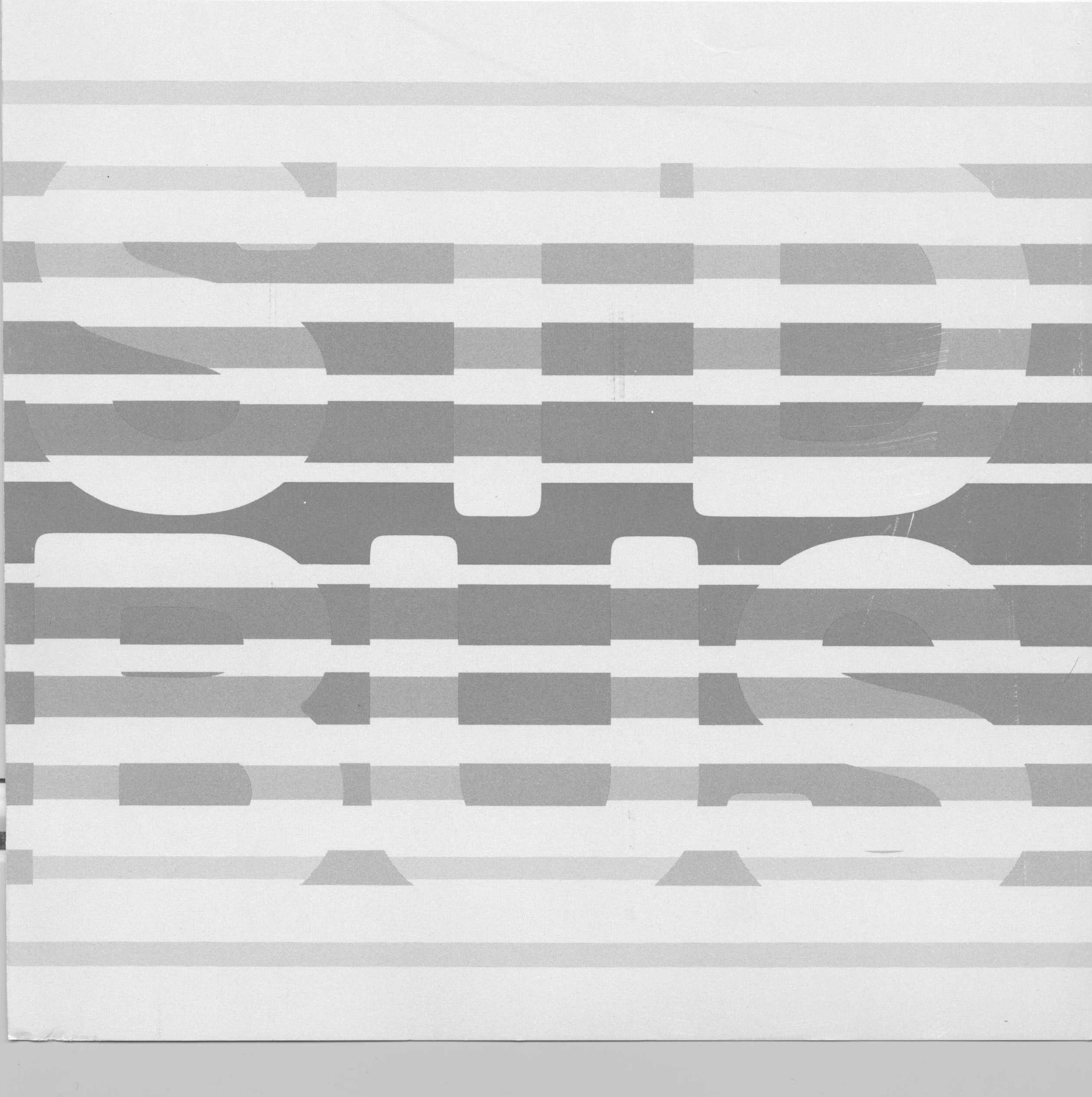
PRO-LOG
CORPORATION

STD 7000

7303
Keyboard/Display Card
USER'S MANUAL



7303
KEYBOARD/DISPLAY CARD
USER'S MANUAL

PRO-LOG
CORPORATION

11/80

FOREWORD

This manual explains how to use Pro-Log's 7303 Keyboard/Display Card. It is structured to reflect the answers to basic questions you, the user, might ask yourself about the 7303. We welcome your suggestions on how we can improve our instructions.

The 7303 is part of Pro-Log's Series 7000 STD BUS hardware. Our products are modular, and designed and built with second-sourced parts that are industry standards. They provide an industrial manager with the means of utilizing his own people to control the design, production, and maintenance of the company's products that use STD BUS hardware.

Pro-Log supports its products with thorough and complete documentation. Also, we teach courses on how to design with, and use, microprocessors and the STD BUS products.

You may find the following Pro-Log documents useful in your work: *Microprocessor User's Guide,* and the *Series 7000 STD BUS Technical Manual.* If you would like a copy of these documents, please write to us on your company letterhead.

Contents

Contents (continued)

Figures

Figures (continued)

Figures (continued)

SECTION 1
Purpose and Main Features

The 7303 is a general purpose, control panel card with data input and display capability (Fig. 1-1). It includes an 8-position alphanumeric display keyboard with 24 program-definable keys plus system reset, an 8-bit binary LED display, and two rocker switches. (See Fig. 1-2 for the block diagram.)

You can use the 7303 in applications where you need a low cost interface for system control, data entry, status display, and operator prompting. Also, the card is useful for system development, testing, and training applications.

The 7303 can be mounted in the first position in a card cage with an open-end panel, on a card extender such as the 7901, or on a 1/8-in. thick panel.

Main Features of the 7303 are:

- **8-position alphanumeric display with ASCII input**
- **24 programmable keys plus reset**
- **Repairable keyboard and replaceable key labels**
- **8-bit binary LED display**
- **2 rocker switches**
- **Simple program control of displays and keys**
- **On-card I/O ports for processor control**
- **Socketed ICs**
- **Single +5V Operation**

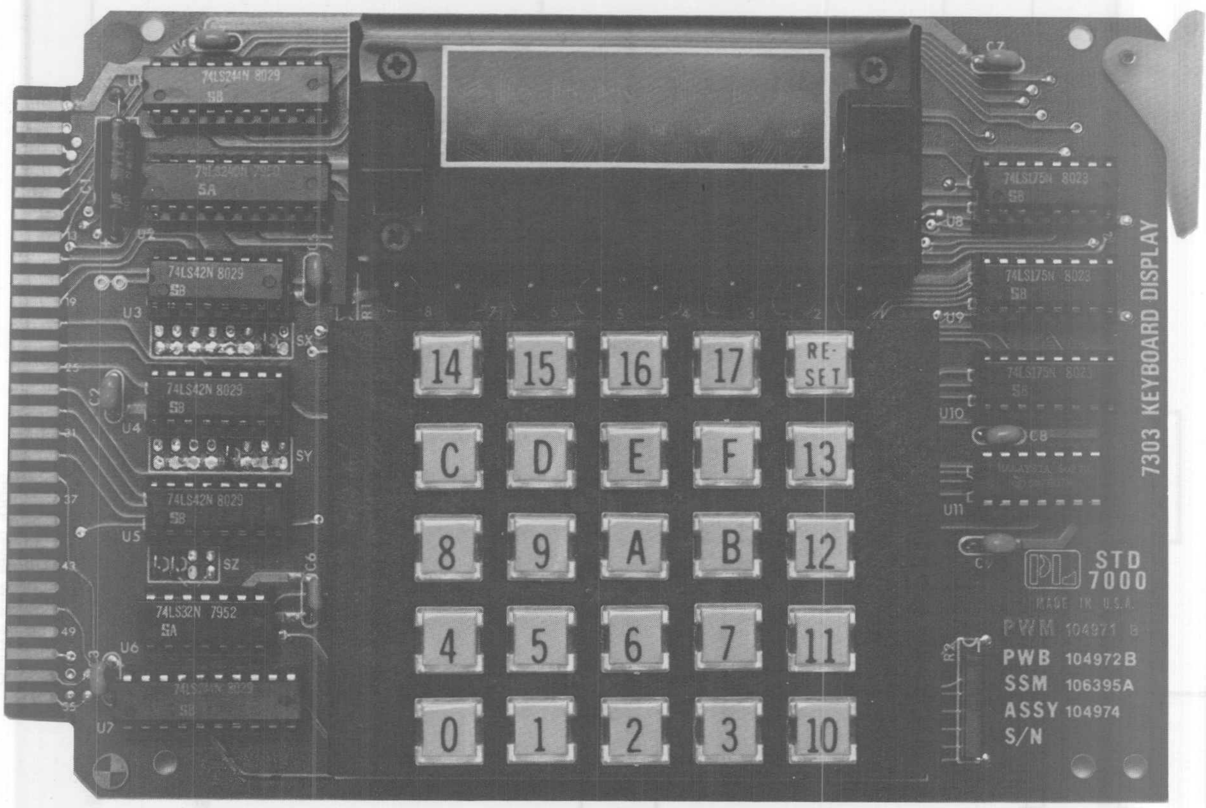
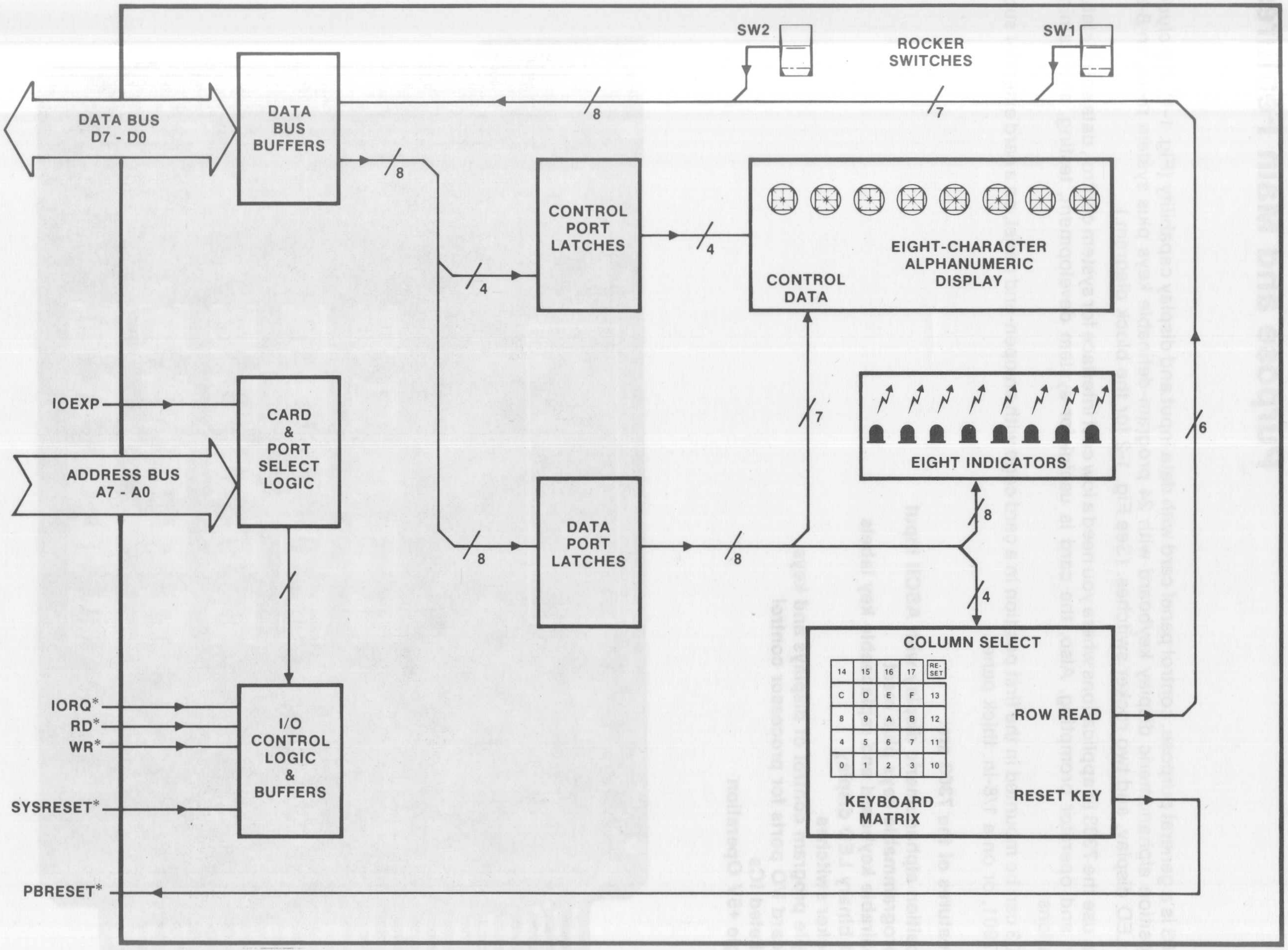


Figure 1-1. 7303 Keyboard/Display Card.



*Active low level logic

Figure 1-2. Block Diagram of the 7303 Keyboard/Display Card.

SECTION 2

Installation and Specifications

The 7303 operates as part of an STD BUS card rack system. You can plug it directly into the STD BUS backplane (Fig. 2-1) or extend it from the motherboard with a 7901 card extender, or equivalent. In this configuration, the card is mapped at processor I/O port addresses.

Insert the card in the left-most socket (viewed from the card ejector end of the rack) of a card cage that has the left end plate open.

Insert a 7901 card extender in any card slot and plug the 7303 into the card extender. In this position, the 7303 clears the other cards and is accessible.

If you mount the 7303 remotely from the card rack, you will need buffering between the card rack and the 7303. A suitable method is to operate the card as an I/O load driven by input and output ports, rather than as an I/O mapped processor-backplane load. For more information, see Pro-Log's *Application Note PLAN 131* (Appendix A).

I/O Mapped Card Addressing

In its normal operation, the 7303 is addressed directly by the processor card. The 7303's input and output ports respond to single read and write instructions executed in the processor's operating program. The 7303 is enabled when a jumper-selected combination of address lines A0 through A7 is present, and when the following control lines are active: IORQ*, IOEXP, and either RD* or WR*.

The 7303 occupies two consecutive I/O addresses regardless of its mapping assignment. The card is shipped with the control port mapped at D1 and the data port mapped at D0. You may retain these addresses or change them by moving the installed jumper wires. By using D0 and D1, the preferred addresses, you can easily adapt standard Pro-Log software. While the card's port addresses are generally arbitrary, they must differ from all other I/O port addresses in the system. If they do not differ, multiple cards will respond to the same READ instruction, resulting in BUS contention.

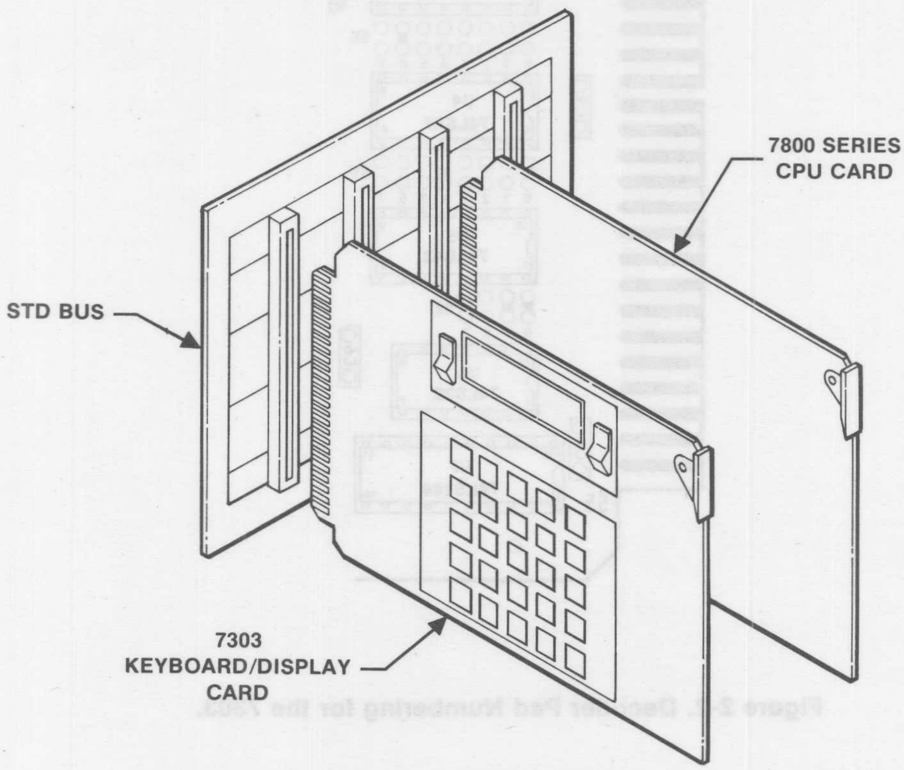


Figure 2-1. I/O Mapped Operation in Local Card Rack.

Changing the Port Addresses

Locate decoders U3, U4, and U5 (74LS42) next to the STD BUS edge connector. Each decoder device has a dual row of pads that form decoder output select matrices. Make one (and only one) connection to each of the matrices next to U3 and U4, and two connections next to U5.

The decoder pad numbering (Fig. 2-2) shows the numbering of the pads next to the decoder chips on the 7303. Also shown are the jumpers (at X6, Y4, Z0, Z1) that produce the hexadecimal port address D0 and D1, the selection made when the card is shipped.

The I/O address mapping and jumper selection for two addresses per card is shown in Figs. 2-3 and 2-4. It indicates where to place jumper straps to obtain any port address in the 00-FF hexadecimal range. Using the 2-digit hexadecimal port addresses desired, find the hexadecimal port addresses along the vertical axis, and read the corresponding strap positions from Fig. 2-4. For example, port address D0 and D1 are obtained by connecting jumpers at X6, Y4, Z0, and Z1. This is the preferred address and is shown on the table by the shaded area.

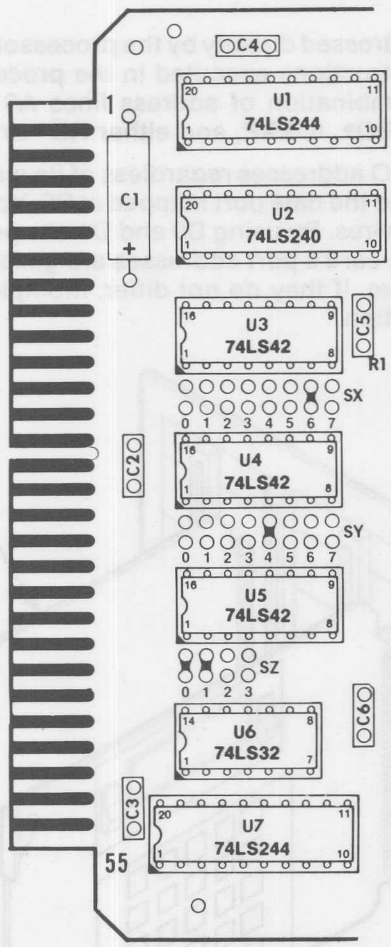


Figure 2-2. Decoder Pad Numbering for the 7303.

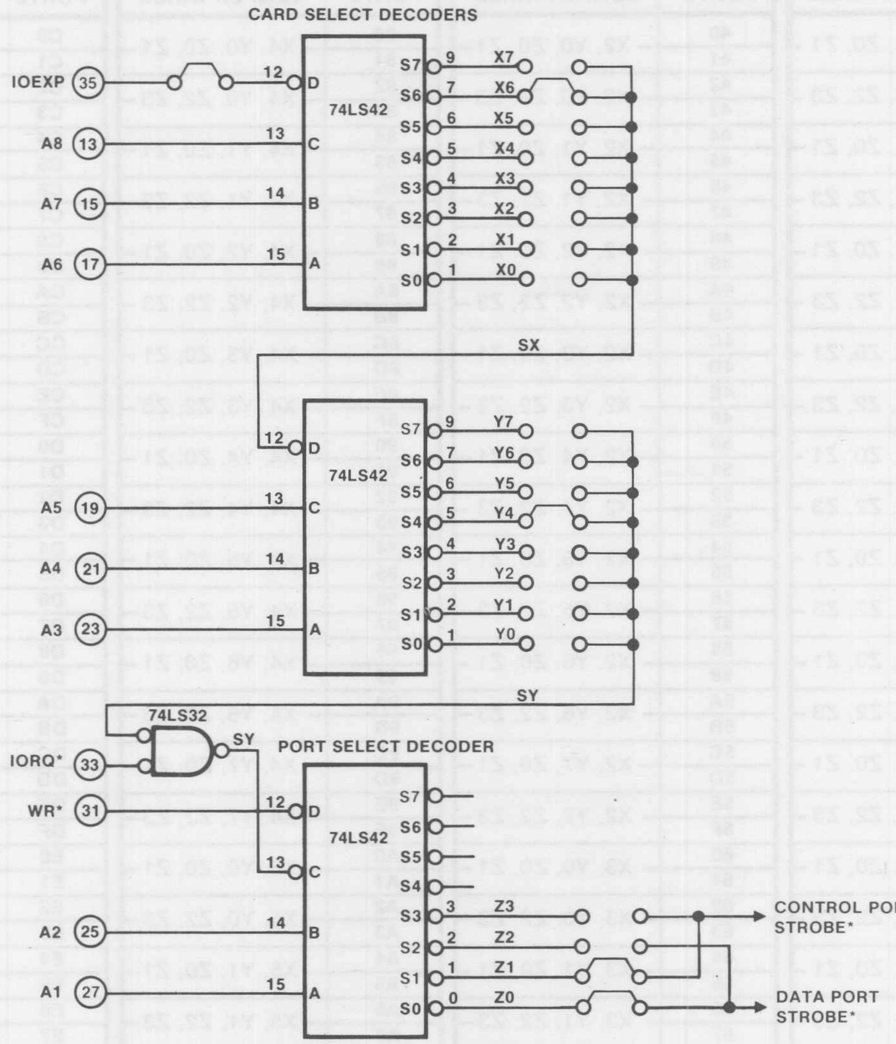


Figure 2-3. 7303 I/O Address Decoder and Schematic for 2 Addresses Per Card
(shown mapped at D0 and D1, the preferred card address).

PORTS	JUMPER WIRES	PORTS	JUMPER WIRES	PORTS	JUMPER WIRES	PORTS	JUMPER WIRES
00 01	X0, Y0, Z0, Z1	40 41	X2, Y0, Z0, Z1	80 81	X4, Y0, Z0, Z1	C0 C1	X6, Y0, Z0, Z1
02 03	X0, Y0, Z2, Z3	42 43	X2, Y0, Z2, Z3	82 83	X4, Y0, Z2, Z3	C2 C3	X6, Y0, Z2, Z3
04 05	X0, Y1, Z0, Z1	44 45	X2, Y1, Z0, Z1	84 85	X4, Y1, Z0, Z1	C4 C5	X6, Y1, Z0, Z1
06 07	X0, Y1, Z2, Z3	46 47	X2, Y1, Z2, Z3	86 87	X4, Y1, Z2, Z3	C6 C7	X6, Y1, Z2, Z3
08 09	X0, Y2, Z0, Z1	48 49	X2, Y2, Z0, Z1	88 89	X4, Y2, Z0, Z1	C8 C9	X6, Y2, Z0, Z1
0A 0B	X0, Y2, Z2, Z3	4A 4B	X2, Y2, Z2, Z3	8A 8B	X4, Y2, Z2, Z3	CA CB	X6, Y2, Z2, Z3
0C 0D	X0, Y3, Z0, Z1	4C 4D	X2, Y3, Z0, Z1	8C 8D	X4, Y3, Z0, Z1	CC CD	X6, Y3, Z0, Z1
0E 0F	X0, Y3, Z2, Z3	4E 4F	X2, Y3, Z2, Z3	8E 8F	X4, Y3, Z2, Z3	CE CF	X6, Y3, Z2, Z3
10 11	X0, Y4, Z0, Z1	50 51	X2, Y4, Z0, Z1	90 91	X4, Y4, Z0, Z1	D0 D1	X6, Y4, Z0, Z1
12 13	X0, Y4, Z2, Z3	52 53	X2, Y4, Z2, Z3	92 93	X4, Y4, Z2, Z3	D2 D3	X6, Y4, Z2, Z3
14 15	X0, Y5, Z0, Z1	54 55	X2, Y5, Z0, Z1	94 95	X4, Y5, Z0, Z1	D4 D5	X6, Y5, Z0, Z1
16 17	X0, Y5, Z2, Z3	56 57	X2, Y5, Z2, Z3	96 97	X4, Y5, Z2, Z3	D6 D7	X6, Y5, Z2, Z3
18 19	X0, Y6, Z0, Z1	58 59	X2, Y6, Z0, Z1	98 99	X4, Y6, Z0, Z1	D8 D9	X6, Y6, Z0, Z1
1A 1B	X0, Y6, Z2, Z3	5A 5B	X2, Y6, Z2, Z3	9A 9B	X4, Y6, Z2, Z3	DA DB	X6, Y6, Z2, Z3
1C 1D	X0, Y7, Z0, Z1	5C 5D	X2, Y7, Z0, Z1	9C 9D	X4, Y7, Z0, Z1	DC DD	X6, Y7, Z0, Z1
1E 1F	X0, Y7, Z2, Z3	5E 5F	X2, Y7, Z2, Z3	9E 9F	X4, Y7, Z2, Z3	DE DF	X6, Y7, Z2, Z3
20 21	X1, Y0, Z0, Z1	60 61	X3, Y0, Z0, Z1	A0 A1	X5, Y0, Z0, Z1	E0 E1	X7, Y0, Z0, Z1
22 23	X1, Y0, Z2, Z3	62 63	X3, Y0, Z2, Z3	A2 A3	X5, Y0, Z2, Z3	E2 E3	X7, Y0, Z2, Z3
24 25	X1, Y1, Z0, Z1	64 65	X3, Y1, Z0, Z1	A4 A5	X5, Y1, Z0, Z1	E4 E5	X7, Y1, Z0, Z1
26 27	X1, Y1, Z2, Z3	66 67	X3, Y1, Z2, Z3	A6 A7	X5, Y1, Z2, Z3	E6 E7	X7, Y1, Z2, Z3
28 29	X1, Y2, Z0, Z1	68 69	X3, Y2, Z0, Z1	A8 A9	X5, Y2, Z0, Z1	E8 E9	X7, Y2, Z0, Z1
2A 2B	X1, Y2, Z2, Z3	6A 6B	X3, Y2, Z2, Z3	AA AB	X5, Y2, Z2, Z3	EA EB	X7, Y2, Z2, Z3
2C 2D	X1, Y3, Z0, Z1	6C 6D	X3, Y3, Z0, Z1	AC AD	X5, Y3, Z0, Z1	EC ED	X7, Y3, Z0, Z1
2E 2F	X1, Y3, Z2, Z3	6E 6F	X3, Y3, Z2, Z3	AE AF	X5, Y3, Z2, Z3	EE EF	X7, Y3, Z2, Z3
30 31	X1, Y4, Z0, Z1	70 71	X3, Y4, Z0, Z1	B0 B1	X5, Y4, Z0, Z1	F0 F1	X7, Y4, Z0, Z1
32 33	X1, Y4, Z2, Z3	72 73	X3, Y4, Z2, Z3	B2 B3	X5, Y4, Z2, Z3	F2 F3	X7, Y4, Z2, Z3
34 35	X1, Y5, Z0, Z1	74 75	X3, Y5, Z0, Z1	B4 B5	X5, Y5, Z0, Z1	F4 F5	X7, Y5, Z0, Z1
36 37	X1, Y5, Z2, Z3	76 77	X3, Y5, Z2, Z3	B6 B7	X5, Y5, Z2, Z3	F6 F7	X7, Y5, Z2, Z3
38 39	X1, Y6, Z0, Z1	78 79	X3, Y6, Z0, Z1	B8 B9	X5, Y6, Z0, Z1	F8 F9	X7, Y6, Z0, Z1
3A 3B	X1, Y6, Z2, Z3	7A 7B	X3, Y6, Z2, Z3	BA BB	X5, Y6, Z2, Z3	FA FB	X7, Y6, Z2, Z3
3C 3D	X1, Y7, Z0, Z1	7C 7D	X3, Y7, Z0, Z1	BC BD	X5, Y7, Z0, Z1	FC FD	X7, Y7, Z0, Z1
3E 3F	X1, Y7, Z2, Z3	7E 7F	X3, Y7, Z2, Z3	BE BF	X5, Y7, Z2, Z3	FE FF	X7, Y7, Z2, Z3

Shading denotes as-shipped configuration.

Figure 2-4. Jumpers Required for 7303 Port Address Mapping.

The jumpers installed at the time of manufacture may be removed and installed at different locations, implementing different port addresses. The preferred method of removing jumpers that have been soldered to the board is to first cut the jumper in half, then unsolder each half individually and discard. Remaining solder should then be removed from the holes and new jumpers installed at the appropriate locations according to the following procedure.

> NOTE
>
> On some early 7303 cards, circuit traces were used instead of wire jumpers to implement ports D0 and D1. In such cases, cut the jumper trace and remove it from the board with a sharp knife, taking care not to damage the board or any other traces; then proceed to install the new jumper(s).

Alternatives to Soldered Wire Jumpers

If occasional or frequent changes in address mapping jumpers are anticipated, remove the wire jumpers and populate the jumper pads with 0.025-in. square posts, which are available individually and in single and double strips corresponding to the 0.100-in. grid jumper pad spacing on the card. The posts may then be connected by wirewrap or by jumper clips available from several sources. Check the height above the board that these parts may protrude, in order to avoid interference with adjacent cards. The recommended wirewrap square post for SX and SY is AMP No. 87215-5, or equivalent. For SZ, it is AMP No. 87215-1, or equivalent. The recommended jump clip is AMP No. 530153-2, or equivalent.

Electrical and Environmental Specifications

SYMBOL	PARAMETER	RECOMMENDED OPERATING LIMITS			ABSOLUTE NONOPERATING LIMITS		
		MIN	TYP	MAX	MIN	MAX	UNIT
Vcc	Supply voltage	4.75	5.00	5.25	0.0	5.50	V
T_A	Free air temperature	0	25	55	0	55	°C
R_H	Humidity [a]	5		95	0	95	%RH

[a] Noncondensing.

Figure 2-5. Electrical Specifications - 7303 Keyboard/Display Card.

SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT
Icc	STD BUS supply current[a]		300	500	mA
—	STD BUS input load		See Fig. 2-7	See Fig. 2-7	
—	STD BUS output drive		See Fig. 2-7	See Fig. 2-7	

[a] All segments driven.

Figure 2-6. STD BUS Electrical Characteristics over Recommended Operating Limits.

MNEMONIC	INPUT (LSTTL LOADS)	OUTPUT (LSTTL DRIVE)	PIN NUMBER	PIN NUMBER	OUTPUT (LSTTL DRIVE)	INPUT (LSTTL LOADS)	MNEMONIC
+5V	VCC		2	1		VCC	+5V
GROUND	GND		4	3		GND	GROUND
-5V			6	5			-5V
D7	1	55	8	7	55	1	D3
D6	1	55	10	9	55	1	D2
D5	1	55	12	11	55	1	D1
D4	1	55	14	13	55	1	D0
A15			16	15		1	A7
A14			18	17		1	A6
A13			20	19		1	A5
A12			22	21		1	A4
A11			24	23		1	A3
A10			26	25		1	A2
A9			28	27		1	A1
A8			30	29		1	A0
RD*	1		32	31		1	WR*
MEMRQ*			34	33		1	IORQ*
MEMEX			36	35		1	IOEXP
MCSYNC*			38	37			REFRESH*
STATUS 0*			40	39			STATUS 1*
BUSRQ*			42	41			BUSAK*
INTRQ*			44	43			INTAK*
NMIRQ*			46	45			WAITRQ*
PBRESET*		OUT	48	47		1	SYSRESET*
CNTRL*			50	49			CLOCK*
PCI	IN		52	51	OUT		PCO
AUX GND			54	53			AUX GND
AUX -V			56	55			AUX +V

* Active low-level logic

Figure 2-7. Edge Connector Pins for the 7303.

Figure 2-8 shows the timing requirements that must be observed by the 7303's operating software. T_{B1} and T_{B2} define the uncertainty period for input port data after a mechanical key or switch opens or closes. Figure 2-9 defines the other data parameters listed below.

SYMBOL	PARAMETER	FROM	TO	MIN	MAX	UNIT
T_{B1}	**Key bounce**	Key depressed or released	Key data stable		15	µs
T_{B2}	**Rocker bounce**	Switch closed or opened	Switch data stable		15	µs
T_{S1}	**Data setup**	ASCII data	Position pulse	1.2		µs
T_{S2}	**Write setup**	Position address	Write pulse	0.6		µs
T_W	**Write width**	Write pulse active	Write pulse inactive	1.1		µs
T_H	**Write hold**	Write pulse	Invalid data address	0.5		µs

Figure 2-8. Switching Characteristics over Recommended Operating Limits—7303 Card.

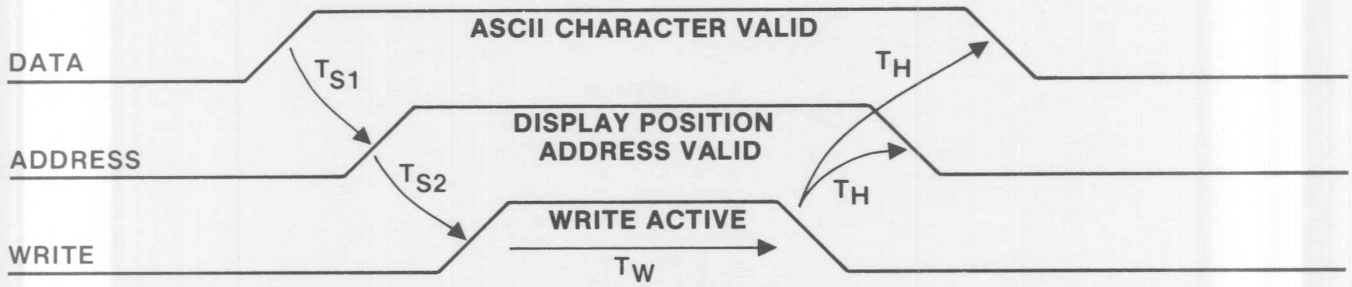


Figure 2-9. 7303 Alphanumeric Display Timing Waveforms.
(Note: Waveforms illustrate program values. WRITE is low level active in hardware.)

Mechanical Specifications

The 7303's storage and nonoperating temperature range is limited to 0 to 55°C.

The 7303 meets all general mechanical specifications of the STD BUS except for component height, which is 0.95 in. (2.14 cm) maximum. If you use the 7303 as an interface card, install it in one of two ways that allow you access to the component side of the card, utilizing a single slot in the card rack.

SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT
—	**Key life expectancy**	3×10^6			Operations
—	**Component height**			0.95	in.

Figure 2-10. Mechanical Characteristics over Recommended Operating Limits—7303 Card.

SECTION 3
Operation and Programming

The 7303, as a general-purpose control panel card, operates as part of the STD BUS card rack system. You can use the 7303 for system control, data entry, status display, and operator prompting in low-cost interface applications. The 7303 can also be used for system development, testing, and training.

The 7303's operator interface consists of an 8-position alphanumeric display; 24 program-definable keys plus a fixed-function reset key that resets the systems's processor card; an 8-bit binary LED display; and two rocker switches. This section shows how each of these elements works and how they are programmed. Actual program examples are found in Section 4.

Figure 3-1 shows the physical layout of the 7303's switches and indicators. It also shows the display position numbers (7-0), the numeric values of the keys in hexadecimal (0-17), and the rocker switch numbers (S1 and S2). These designations are important when programming the 7303, and you will probably want to refer back to Fig. 3-1 while reading the rest of this section.

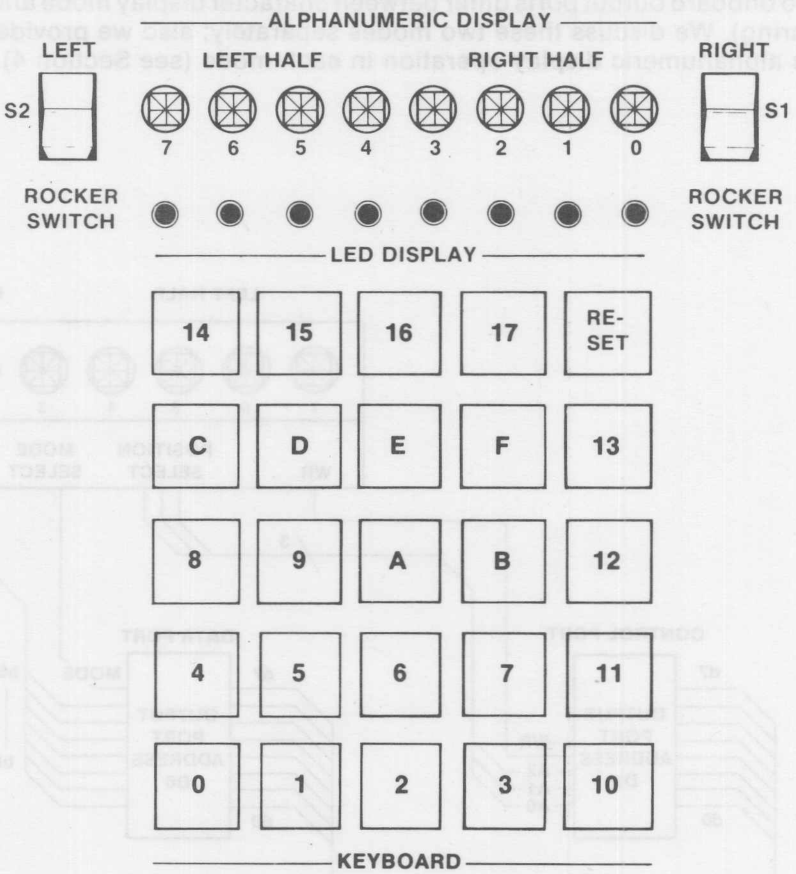


Figure 3-1. 7303 Keyboard/Display.

Alphanumeric Display

The display consists of eight, 16-segment alphanumeric positions. Each position displays any character from the 64-character ASCII set. It can also display a cursor character (all segments on). Each display position has an ASCII character memory and a separate cursor memory. These separate memories allow the cursor to be displayed and removed without altering the ASCII character memory. Each display position is randomly addressable.

Two onboard output ports drive the display (Fig. 3-2). The display's operation is controlled by program manipulation of the output bits from these ports. The ports provide the display with data, addressing, and control signals, giving the program random access to any of the eight display positions.

You can program each display position in either of two modes: **character** or **cursor.** By flashing the cursor (all segments on) alternately with another character, you can draw attention to one or more of the display positions. Also, you can use the cursor as a lamp test. The display can have any combination of characters and cursors present.

In the **character display mode,** you can load each display position with any of the characters shown in Fig. 3-3. Use the SPACE character to blank the position. Note that the display uses 7-bit ASCII code. Each display position has its own ASCII character memory, ASCII-to-16-segment decoder, and lamp drivers.

In the **cursor display mode,** each display position can show the cursor character, and each position has a separate cursor memory in addition to its character memory. Since setting the cursor-on memory bit does not alter the content of the ASCII character memory, you can flash the cursor and an ASCII character alternately by setting and clearing the cursor memory.

The functions of the two onboard output ports differ between character display mode and cursor display mode (including display clearing). We discuss these two modes separately; also we provide separate subroutine modules for the 7303's alphanumeric display operation in each mode (see Section 4).

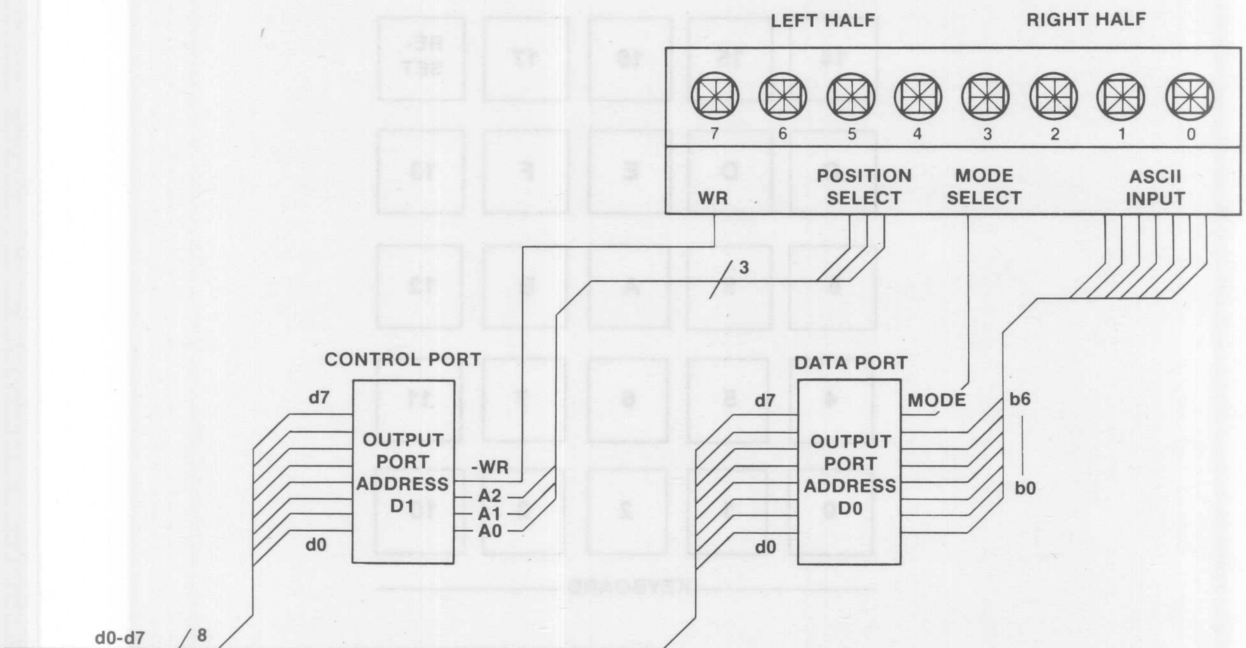


Figure 3-2. Alphanumeric Display Programming Model for the 7303.

ASCII CHAR	HEX CODE	ASCII CHAR	HEX CODE	ASCII CHAR	HEX CODE	ASCII CHAR	HEX CODE
SPACE	A0	0	B0	@	C0	P	D0
!	A1	1	B1	A	C1	Q	D1
”	A2	2	B2	B	C2	R	D2
#	A3	3	B3	C	C3	S	D3
$	A4	4	B4	D	C4	T	D4
%	A5	5	B5	E	C5	U	D5
&	A6	6	B6	F	C6	V	D6
’	A7	7	B7	G	C7	W	D7
(	A8	8	B8	H	C8	X	D8
)	A9	9	B9	I	C9	Y	D9
*	AA	:	BA	J	CA	Z	DA
+	AB	;	BB	K	CB	[	DB
,	AC	<	BC	L	CC	\	DC
-	AD	=	BD	M	CD	]	DD
.	AE	>	BE	N	CE	^	DE
/	AF	?	BF	O	CF	—(Note)	DF

Note: Underscore.

Figure 3-3. Hexadecimal Values of ASCII Characters.

Initialization: Reset Characteristics. The 7303's SYSRESET* input clears its output ports but does not clear the alphanumeric display or its character and cursor memories. If SYSRESET* occurs while the program is changing the content of the alphanumeric display, the content may be altered unpredictably. Therefore, make sure you restore or clear the alphanumeric display after a system reset.

Also, after power-on, the display's content is unpredictable. So initialization by a programmed instruction sequence is generally needed soon after power-on. To blank the display, load the SPACE character (ASCII /hexadecimal A0) in each display position. Note that a separate instruction sequence is required to clear the cursors.

ASCII Character Set. The 7303 can display 64 different characters. These characters, and the hexadecimal code to produce each one, are given in Fig. 3-3.

To use this figure, identify the character you wish displayed. The code to the right of the character is a two-digit hexadecimal number that uniquely identifies the character. For the 64 characters that the 7303 can display, the codes range from A0 through DF. For example: the hexadecimal code for the SPACE character is A0, for the number 3 it is B3, and for the letter M it is CD.

The use of hexadecimal codes not listed in the figure results in either a blanked display position (if bit 7 of the code is 1), or undefined cursor activity (if bit 7 is 0).

NOTE on Port Addresses

Section 2 shows how you can remap the 7303's address decoders to allow the card to occupy any two consecutive port addresses in the 00-FF hexadecimal range.

The 7303 is shipped with port addresses D0 and D1 selected by jumper wires, and all of the explanation of the card's operation and programming in this section assumes that these addresses remain connected.

If you elect to remap the 7303, regard the onboard ports as the Data Port and the Control Port (ports D0 and D1, respectively).

Output Port Bit Assignments for Character Mode

Data Port. Output port D0 selects character mode (bit 7 = 1) and specifies one of the 64 ASCII characters to be displayed in bits 0-6. Figure 3-4 shows the bit assignments in the data port for character mode.

DATA BUS	MNEM	DESCRIPTION
d7	MODE	1 = Character mode
d6	b6	MSB
d5	b5	
d4	b4	
d3	b3	7-bit ASCII character
d2	b2	
d1	b1	
d0	b0	LSB

Note: Standard data port address is HEX D0.

Figure 3-4. Data Port Bit Assignments for Character Mode—7303 Card.

Control Port. Output Port D1 selects the alphanumeric display position address (bits 2, 1, 0) and enables the display's WRITE function as shown in Fig. 3-5.

DATA BUS	MNEM	DESCRIPTION
d7	X	Don't care
d6	X	
d5	X	
d4	X	
d3	WR	1 = Write, 0 = Write inhibit
d2	A2	Display position address 0-7
d1	A1	See Fig. 3-6
d0	A0	

Note: Standard control port address is HEX D1.

Figure 3-5. Control Port Bit Assignments for Character Mode—7303 Card.

Figure 3-6 shows the bit patterns required in the control port's bits 2, 1, 0 to address the eight alphanumeric display positions 0-7.

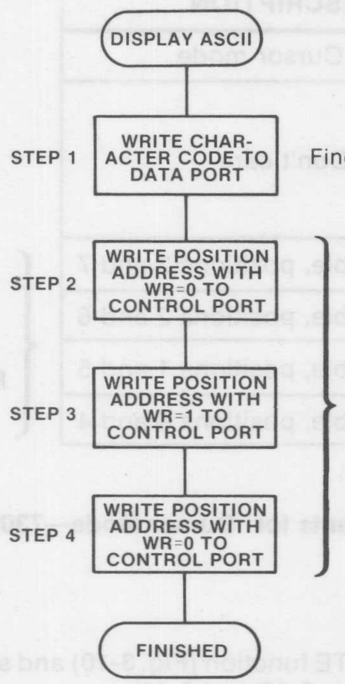
DATA BUS	MNEM	DISPLAY POSITION 7	6	5	4	3	2	1	0
d2	A2	1	1	1	1	0	0	0	0
d1	A1	1	1	0	0	1	1	0	0
d0	A0	1	0	1	0	1	0	1	0

Figure 3-6. Display Position Addressing—7303 Card.

Programming in the Character Display Mode. Causing one of the ASCII characters to appear in one of the 7303's display positions requires four steps in the program. These four steps can be summarized as follows:

1. Output the hexadecimal value of the ASCII character to be displayed (Fig. 3-3) to the 7303's data port (Fig. 3-4).
2. Output the 3-bit address of the display position the character is to occupy (7-0) with the write bit = 0 to the control port (Fig. 3-5).
3. Repeat step 2, but set the write bit = 1.
4. Repeat step 2 (write bit returns to zero, protecting the display).

These steps are summarized as a flow diagram and resulting waveforms in Figs. 3-7 and 3-8 below.



PROGRAM STEP	DISPLAY POSITION 7	6	5	4	3	2	1	0	COMMENT
Step 2	07	06	05	04	03	02	01	00	Write = 0
Step 3	0F	0E	0D	0C	0B	0A	09	08	Write = 1
Step 4	07	06	05	04	03	02	01	00	Write = 0

Program Values for Steps 2, 3, 4.

Figure 3-7. Flow Diagram of Character Mode Events for the 7303.

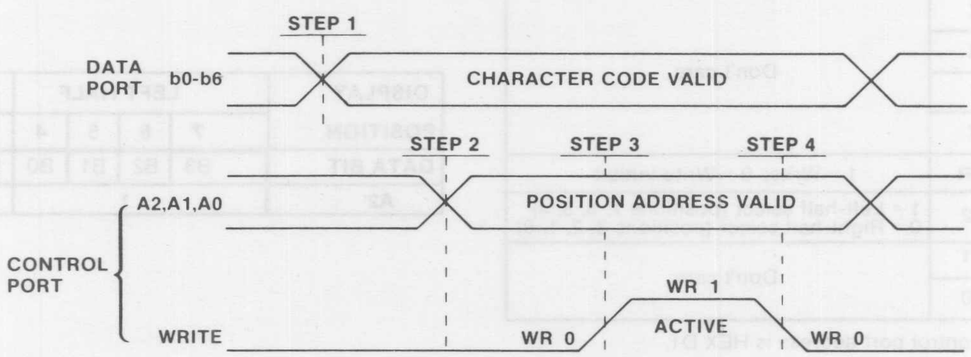


Figure 3-8. Character Mode Timing Waveforms for the 7303.
(Note: Waveforms illustrate program values. WR is low active in hardware.)

Cursor Mode

Once a valid ASCII character is loaded into the display position's ASCII memory, the position can display the cursor character. Note that ASCII characters must be displayed before the cursor can be displayed; the SPACE character satisfies this requirement.

Output Port Bit Assignments for Cursor Mode

Cursor mode and character mode share the same output ports, but the bit functions differ between the two modes.

Data port. Output port D0 selects cursor mode (bit 7 = 0). Bits 0, 1, 2, 3 specify the cursor on/off state for four display positions at a time. Either the right half of the displays (positions 0, 1, 2, 3) or the left-half of the displays (positions 4, 5, 6, 7) can be addressed in one operation. Figure 3-9 shows the data port bit assignments for cursor mode.

DATA BUS	MNEM	DESCRIPTION	
d7	MODE	0 = Cursor mode	
d6	b6	Don't care	
d5	b5		
d4	b4		
d3	b3	Cursor enable, positions 3 and 7	Set bit = 1 to display cursor. Reset bit = 0 to remove cursor.
d2	b2	Cursor enable, positions 2 and 6	
d1	b1	Cursor enable, positions 1 and 5	
d0	b0	Cursor enable, positions 0 and 4	

Note: Standard data port address is HEX D0.

Figure 3-9. Data Port Bit Assignments for Cursor Mode—7303 Card.

Control Port. Output port D1 controls the display's WRITE function (Fig. 3-10) and selects between the right-hand four displays and the left-hand four displays (Figs. 3-10 and 3-11).

DATA BUS	MNEM	DESCRIPTION
d7	X	Don't care
d6	X	
d5	X	
d4	X	
d3	WR	1 = Write; 0 = Write inhibit
d2	A2	1 = Left-half select (positions 7, 6, 5, 4) 0 = Right-half select (positions 3, 2, 1, 0)
d1	A1	Don't care
d0	A0	

Note: Standard control port address is HEX D1.

Figure 3-10. Control Port Bit Assignments for Cursor Mode—7303 Card.

DISPLAY	LEFT HALF				RIGHT HALF			
POSITION	7	6	5	4	3	2	1	0
DATA BIT	B3	B2	B1	B0	B3	B2	B1	B0
A2	1				0			

Figure 3-11. Left/Right Display Position Group Select for Cursor Mode—7303 Card.

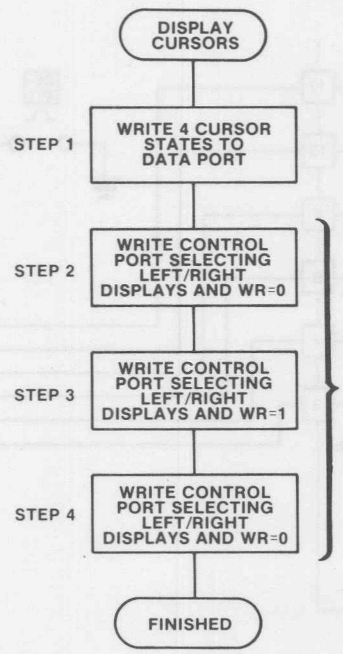
Programming in the Cursor Display Mode. With a valid ASCII character loaded to a display position, the cursor character can also be displayed in that position. When the cursor is removed, the same ASCII character will reappear.

Cursor characters can be turned on or off in any combination, in groups of four display positions (right half = positions 0, 1, 2, 3 and left half = positions 4, 5, 6, 7). Controlling all eight cursors requires two separate operations.

Setting/clearing the left-half or right-half cursor memories requires four steps in the program:

1. Output the desired states of four of the cursors to the data port (Fig. 3-9).
2. Output the left/right select bit with write = 0 to the control port (Fig. 3-10).
3. Repeat step 2, but set the write bit = 1.
4. Repeat step 2 (write bit returns to zero, protecting the display).

These steps are summarized as a flow diagram and resulting waveforms in Figs. 3-12 and 3-13 below.



	DISPLAY POSITION		
PROGRAM STEP	LEFT HALF	RIGHT HALF	COMMENT
Step 2	04	00	Write = 0
Step 3	0C	08	Write = 1
Step 4	04	00	Write = 0

Program Values for Steps 2, 3, 4.

Figure 3-12. Flow Diagram of Cursor Mode Events for the 7303.

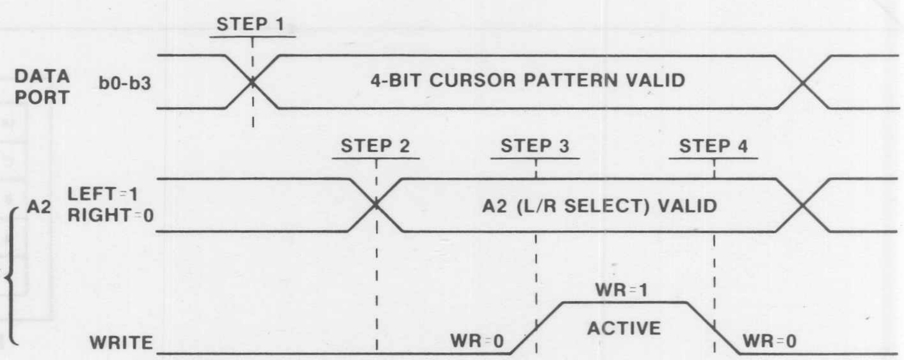


Figure 3-13. Cursor Mode Timing Waveforms for the 7303.
(Note: Waveforms illustrate program values. WR is low active in hardware.)

Keyboard

The keyboard consists of a RESET key and 24 program-definable keys (Fig. 3-14).

The **RESET key** is not programmable. When pressed, it grounds the 7303's PBRESET* output to the STD BUS backplane. This signal is provided to reset the system processor card, which responds by generating SYSRESET*. SYSRESET* is an input to the 7303 card, which resets the 7303's output ports. The exact characteristics of the SYSRESET* signal depend on the processor card in use.

The **24 program-definable keys** are wired in a 4 x 6 switch matrix. The four columns (vertical axis) are driven by the data port (output D0 port bits 0, 1, 2, 3) and the six rows (horizontal axis) are sensed by input port D0 bits 0, 1, 2, 3, 4, 5.

Reading the keyboard is a programmed operation. The program strobes each column of keys in turn, using rotate or shift instructions to move the strobe (a logic "1") from column to column. As each column is strobed, the program reads the input port to see if a switch closure has connected the strobe bit to the input port. If so, both key coordinates are now known (the program generated the column value and the input port read the row value), so that the value of the key can be computed. If not, the program steps the strobe to the next column and repeats the process until a key closure occurs.

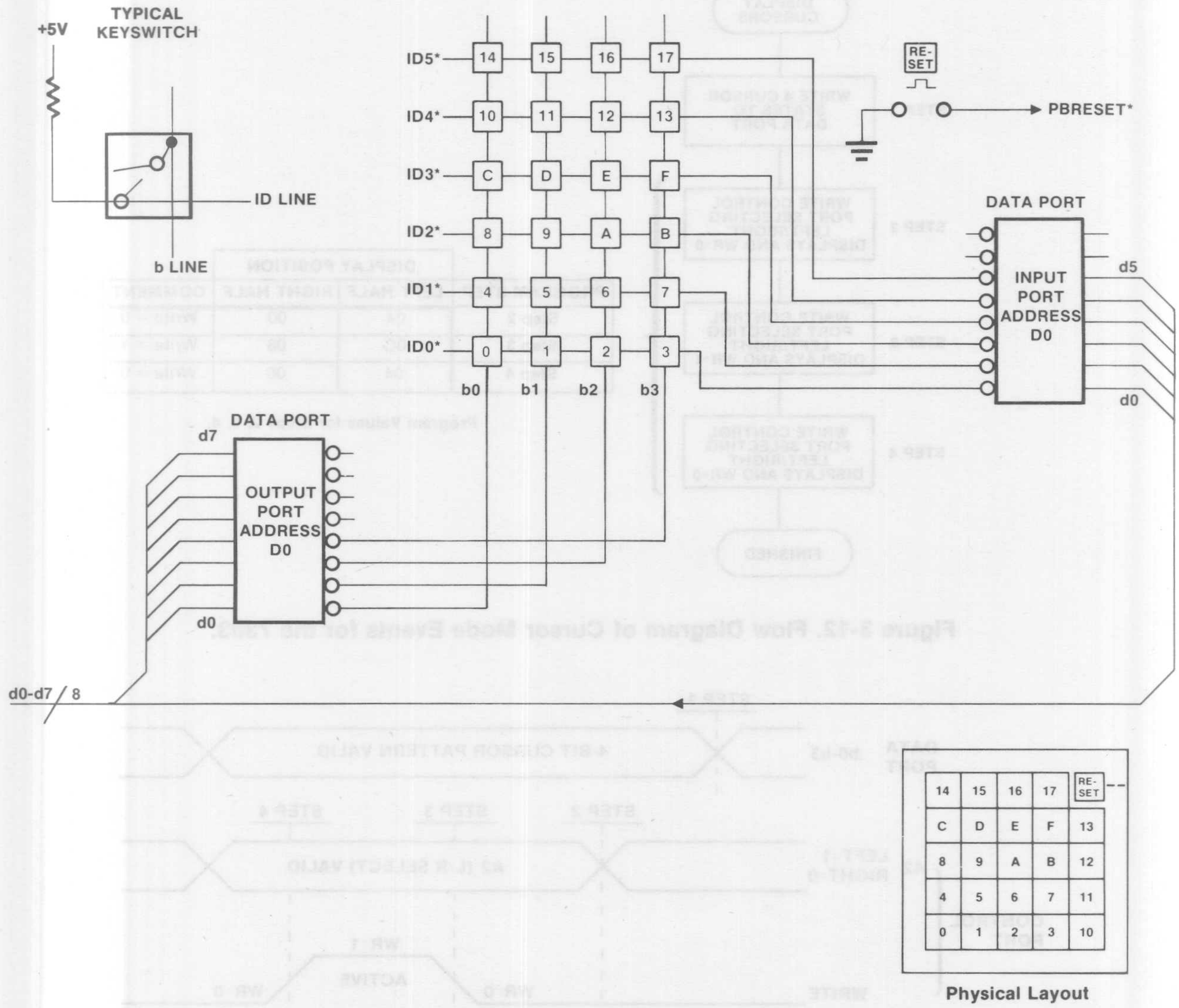


Figure 3-14. Keyboard Programming Model for the 7303.

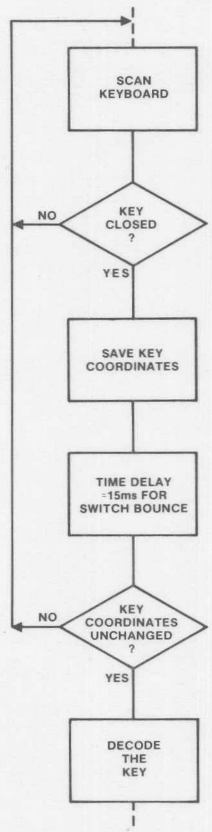
Key Values. The value assigned to a key is an arbitrary, unique identifier that can be derived once the column and row coordinates are known. The (DECODE.KEY) subroutine provided in the 7303's software package in Section 4 uses an algorithm that identifies each key with a hexadecimal number in the 00-17 range. The 7303 is shipped with key labels that show the value that will be generated by the (DECODE.KEY) subroutine when the key is pressed.

Frequently, the value associated with a key is meaningless in relation to the application, and the user may wish to rename the key with a more meaningful label. The generalized (DECODE.KEY) subroutine is still used to locate a key closure, but the value returned is decoded a second time to lead to a specific system function. For example, the CALCULATOR program example in Section 4 shows how to use the compare and conditional jump instructions to detect the "11" key and assign it the "CLEAR DISPLAY" system function.

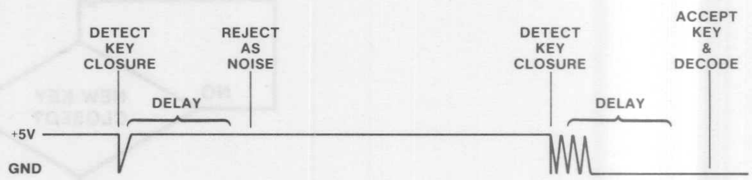
Key Reading Procedures. In addition to simply detecting and decoding a key closure, the program may also be responsible for the following key-control procedures:

1. Differentiate between noise and a genuine key closure.
2. Ignore key-contact bounce when a key closes or opens.
3. React only when the key closes, not when it opens (or vice versa).
4. Avoid multiple responses to the same closure.

Noise and key-contact bounce can be suppressed by programming a double READ with a time delay between the READs as shown in Fig. 3-15.



Example of Program Flow



Voltage Waveform at Row in Key Matrix

Figure 3-15. Programming Key Bounce and Noise Rejection for the 7303.
(Note: This figure illustrates the technique of read/delay/re-read/compare, which allows the program to differentiate between noise and a legitimate key closure, and to pause while the key contacts settle.)

In most instances, it is desirable for the key to be effective when pressed, not when released. Because of the speed of microprocessors, there is also a real possibility that the system might react more than once to the same key closure before the operator can remove his finger (with practice, an operator can deliberately close and release a small pushbutton in about 50ms; however, this represents an absolute minimum and the program should not make assumptions about the operator's characteristics).

The (READ.KEY) subroutine in Section 4 shows how to combine the key decode process with procedural controls to produce reliable, error-free keyboard entries.

The basic assumption in the (READ.KEY) routine is that when the subroutine is entered, the operator's finger is still on the key that was just decoded. The software waits until the operator releases the previous key, then waits again until he presses the next key, then decodes the next key. This technique ensures two important characteristics:

1. The system will react one and only one time to one key closure.
2. The system's reaction will take place immediately after the key is closed and not when it is released.

Figure 3-16 shows a flow diagram of the major events during the (READ.KEY) subroutine.

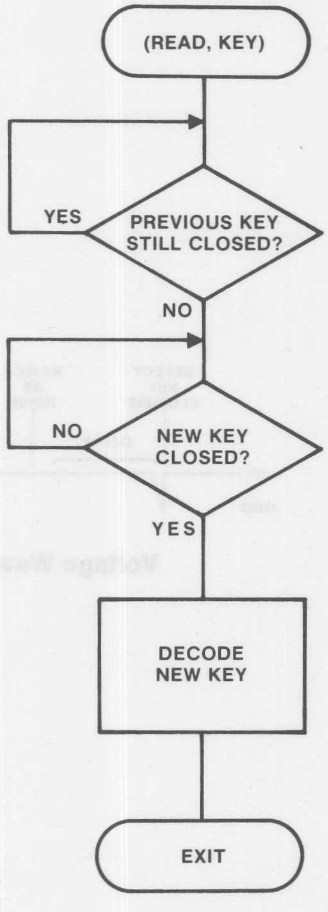


Figure 3-16. Recommended System-Level Keyboard Procedure for the 7303.
(Note: Contact bounce and noise rejection are not shown.)

Binary LED Display

The 8-bit binary LED display (Fig. 3-17) is driven directly by output data port D0—the same output port that strobes the keyboard and supplies ASCII data to the alphanumeric display. When a bit from this port is in the high state, the corresponding LED lights up. The LED display is cleared by the SYSRESET* input.

Because output data port D0 is used in both alphanumeric display and keyboard decoding operations, the binary LEDs change when you address either the display or keyboard. The binary LEDs are useful in training, or in developing programs for the alphanumeric display and keyboard.

You can also use the binary LEDs to display data that is unrelated to the alphanumeric display and keyboard, but when you do:

1. Refresh the binary LED display after any keyboard scan or alphanumeric display operation.
2. Note that the binary LEDs will show dynamic keyboard-scanning activity for as long as a keyboard key is depressed (using the subroutine in Section 4).
3. Do not output binary display information to the LEDs, unless the alphanumeric display's WRITE bit (output port D1, bit 3) is first set to the "0" state to inhibit changes in alphanumeric display.

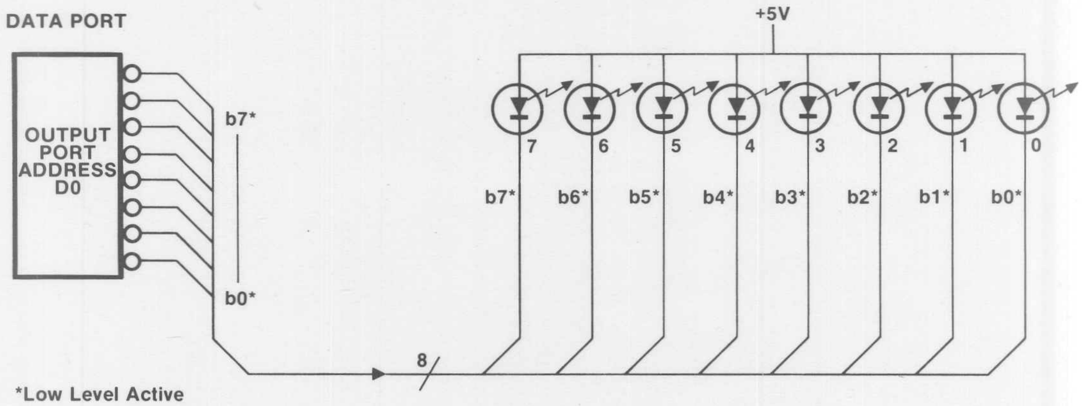


Figure 3-17. Binary LED Display for the 7303.

Rocker Switches

Two rocker-type toggle switches (uncommitted) provide general mode selection. They connect directly to bits 6 and 7 of input port D0, respectively (Fig. 3-18). Their condition (ON or OFF) can be read by the program at any time. Figure 3-19 shows the logic state returned according to switch position. Switch S1 is on the right side of the display and S2 is on the left.

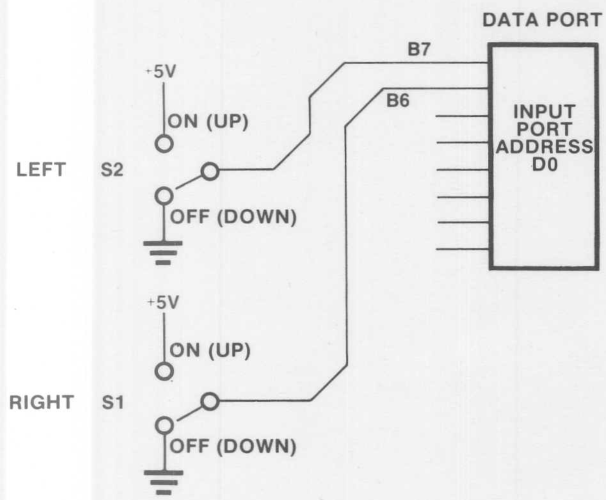


Figure 3-18. Rocker Switches for the 7303.

SWITCH POSITION		ROCKER SWITCH S1 ON (up)	ROCKER SWITCH S1 OFF (down)	ROCKER SWITCH S2 ON (up)	ROCKER SWITCH S2 OFF (down)
Input Port D0	Bit 7	—	—	1	0
(Data Port)	Bit 6	1	0	—	—

Figure 3-19. Rocker Switch Status for the 7303.

SECTION 4
Operating Software

Introduction

This section contains hardware-level subroutine modules with which to operate the display and keyboard. It also includes short programs that may help you in testing or repairing the card, and that illustrate how the subroutines can be linked to work together at system level.

The software in this section can be used without license from Pro-Log. Although tested and believed correct, this software is not represented to be free from errors or copyright infringement, or appropriate for any specific application.

The subroutines are in STD instruction mnemonics, using 8080 assembly codes. They execute in 8080, 8085, Z80, NSC 800, and other code-compatible microprocessor systems. The coding forms are grouped at the end of this section, following the flowcharts.

Flowcharts, which do not refer to microprocessor characteristics, allow the subroutines to be easily adapted to other microprocessor types.

The subroutines are grouped in functional modules. Each module specification describes the module's content, including flowcharts. Individual subroutine specifications give memory, entry, and exit requirements for each path, plus timing, and other necessary information.

Memory Addresses

Full memory addresses are given. They are preferred addresses that allow the subroutines to work with those provided for other Series 7000 STD BUS cards from Pro-Log. The program addresses correspond to the Series 7800 processor cards' onboard ROM/EPROM and RAM sockets.

If your system can not use the memory addresses in the 7303's software package, simply change the memory page addresses, as required, when loading these modules into your system. Memory addresses that **must** be located in RAM are noted on the program coding forms. Other locations are intended for ROM storage, but they can also be executed in RAM.

I/O Port Addresses

The 7303's I/O ports are assigned preferred hexadecimal addresses D0 and D1 for compatibility with other Series 7000 cards. Section 2 shows how to remap these addresses if necessary. This software can be used by simply changing the port addresses when loading the program modules into your system.

Note that each input (IPA) and output (OPA) instruction is extended to three bytes by the addition of a no-operation (NOP) instruction in this software. This allows the user to replace the IPA and OPA instructions with the 3-byte LDAD/STAD instructions, if the 7303 card is memory-mapped (with a memory page address decoder provided by the user on another card to generate the IORQ* signal). Also, the IPA/OPA instructions can be replaced by jump-to-subroutine (JS) instructions for constructing subroutines in RAM, to read/write the 7303's ports. This allows the program to vary the port address, which in turn allows the same software package to be used for several 7303 cards in the same card rack.

Software Package Contents

Figures 4-1 and 4-2 list the demonstration/test programs and subroutines, respectively, in the 7303's software package.

Figure 4-1 lists short, endless-loop operating programs for demonstrating and repairing the 7303. These programs are examples of how the subroutines in the software package can be linked together. Monitor the execution of these programs with an M800 system analyzer and other test equipment to facilitate repair of the 7303, or use them as programming examples or for educational purposes.

PROGRAM NAME	FUNCTION	SEE FIGURE
DISPLAY.DEMO	Uses (BILLBOARD) and (LAMP.TEST) subroutines. Illustrates a technique for displaying a long message on a display with a limited number of positions. Repeats the message "PRO-LOG 7303" twice, tests LED segments, then repeats.	4-56
DISPLAY.SELF	Displays address/data for the 256 memory bytes in memory page 10, which is where the display subroutines are stored. Displays information on the program coding forms in this section, then repeats. Uses (DISP.2.IN.C).	4-57
CALCULATOR	Illustrates how (READ.KEY) and (MESSAGE) can work together with memory manipulation to create a calculator-style data entry, with keystrokes shifted from right to left across the display.	4-58
DISPLAY.TEST	Uses (DISPLAY.8) to step the 7303's display through the entire ASCII character set with each character displayed in sequence in all eight display positions.	4-59
KEY.TEST	Uses (READ.KEY) and (DISP.2.IN.C) to display the 2-digit hex value of each key when the key is pressed. Allows the operator to test each key or to monitor the decode and display processes on the M800 system analyzer.	4-60

( ) Denotes subroutine labels

Figure 4-1. Index of Demonstration and Test Programs for the 7303.

(Note: Because these programs are written as endless loops, it is necessary to reset the system processor to exit from them.)

Figure 4-2 lists the general purpose, hardware-level subroutines provided for operating the 7303. These subroutines allow the user's program to communicate with the 7303 via data "mailboxes" in the processor's internal registers and in RAM, avoiding the need to write port and bit manipulation software.

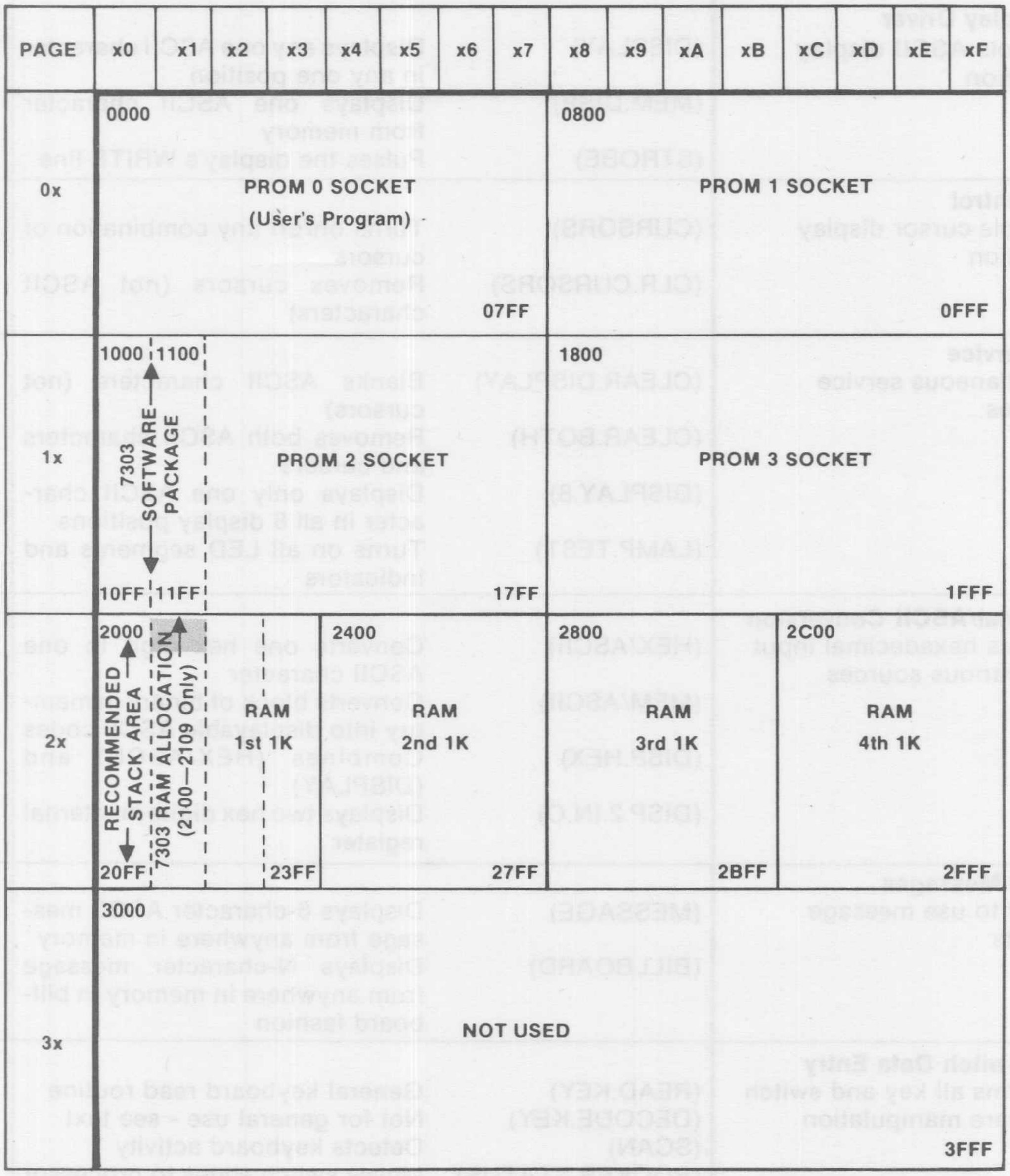
MODULE NAME	SUBROUTINE NAME	FUNCTION	SEE FIGURE
ASCII Display Driver			4-7
Controls ASCII display operation	(DISPLAY)	Displays any one ASCII character in any one position	4-8 & 4-9
	(MEM.DISP)	Displays one ASCII character from memory	4-10 & 4-11
	(STROBE)	Pulses the display's WRITE line	4-12 & 4-13
Cursor Control			4-14
Controls cursor display operation	(CURSORS)	Turns on/off any combination of cursors	4-15 & 4-16
	(CLR.CURSORS)	Removes cursors (not ASCII characters)	4-17 & 4-18
Display Service			4-19
Miscellaneous service routines	(CLEAR.DISPLAY)	Blanks ASCII characters (not cursors)	4-20 & 4-21
	(CLEAR.BOTH)	Removes both ASCII characters and cursors	4-22 & 4-23
	(DISPLAY.8)	Displays only one ASCII character in all 8 display positions	4-24 & 4-25
	(LAMP.TEST)	Turns on all LED segments and indicators	4-26 & 4-27
Hexadecimal/ASCII Conversion			4-28
Accepts hexadecimal input from various sources	(HEX/ASCII)	Converts one hex digit to one ASCII character	4-29 & 4-30
	(MEM/ASCII)	Converts block of binary in memory into displayable ASCII codes	4-31 & 4-32
	(DISP.HEX)	Combines (HEX/ASCII) and (DISPLAY)	4-33 & 4-34
	(DISP.2.IN.C)	Displays two hex digits in internal register	4-35 & 4-36
Formatted Messages			4-37
Ready to use message formats	(MESSAGE)	Displays 8-character ASCII message from anywhere in memory	4-38 & 4-39
	(BILLBOARD)	Displays N-character message from anywhere in memory in billboard fashion	4-40 & 4-41
Key and Switch Data Entry			4-42
Performs all key and switch hardware manipulation	(READ.KEY)	General keyboard read routine	4-43 & 4-44
	(DECODE.KEY)	Not for general use - see text	—
	(SCAN)	Detects keyboard activity	4-45 & 4-46
	(ROCKER.STATUS)	Moves switch states to processor status flags	4-47 & 4-48
Auxiliary Timing			4-49
Inexact delays for display viewing and switch debounce	(DISPLAY.DELAY)	Not for general timing applications - see text	4-50 & 4-51
	(LONG.DELAY)	Not for general timing applications - see text	4-52 & 4-53
	(DEBOUNCE.DELAY)	Not for general timing applications - see text	4-54 & 4-55

( ) Denotes subroutine labels

Figure 4-2. Index of Keyboard and Display Subroutines for the 7303.

Memory Maps

Figures 4-3 through 4-6 are memory maps. Figure 4-3 shows the 16K address space occupied by the Series 7800 processor cards and the location of the 7303 software package within the processor card's memory. Figures 4-4 and 4-5 map the specific subroutines within memory pages 10 and 11 (hexadecimal locations 1000-11FF). Figure 4-6 shows the RAM "mailbox" area within memory page 21 (hexadecimal locations 2100-2109).



NOTES

1. 7801 (8085A) and 7303 (Z80) processor cards have sockets for 8K ROM/PROM (sockets labeled PROM 0 - PROM 3). These cards are shipped with these sockets empty. Also, the cards have sockets for 4K.RAM, and the card is shipped with 1st 1K loaded and 2nd, 3rd, and 4th 1K sockets empty.
2. This map shows the 7303 software loaded in user-supplied PROM 2. Ten locations (2100-2109) in the RAM supplied with the processor card are used by the software. Page 20 (memory addresses 2000-20FF) is recommended for the subroutine return address stack.

Figure 4-3. 16K Memory Map—7303 Software Package in 7801/7803 Processor Card Onboard Memory Sockets.

PAGE ADDRESS 10

LINE	LABEL	LINE	LABEL	LINE	LABEL	LINE	LABEL
00	(MEM.DISP)	40		80	(CLR.CURSORS)	C0	
01		41		81		C1	
02		42		82	(CURSORS)	C2	
03		43		83		C3	
04		44		84		C4	
05		45		85		C5	
06		46		86		C6	
07	(STROBE)	47		87		C7	(DISP.HEX)
08		48		88		C8	
09		49		89		C9	
0A		4A		8A		CA	
0B		4B		8B		CB	
0C		4C		8C		CC	
0D		4D	(DISPLAY.8)	8D		CD	
0E		4E		8E		CE	(DISP.2.N.C)
0F		4F		8F		CF	
10		50		90		D0	
11		51		91		D1	
12		52		92		D2	
13		53		93		D3	
14		54		94		D4	
15		55		95		D5	
16		56		96		D6	
17		57		97		D7	
18		58		98		D8	
19		59		99		D9	
1A	(CLEAR.BOTH)	5A		9A		DA	
1B		5B		9B		DB	
1C		5C		9C		DC	
1D	(CLEAR.DISPLAY)	5D		9D	(HEX/ASCII)	DD	
1E		5E		9E		DE	
1F		5F		9F		DF	
20		60	DEMO MESSAGE	A0		E0	(DISPLAY.DELAY)
21		61		A1		E1	
22		62		A2		E2	
23		63		A3		E3	
24		64		A4		E4	
25		65		A5		E5	(DEBOUNCE.DELAY)
26		66		A6		E6	
27		67		A7		E7	
28		68		A8		E8	
29		69		A9		E9	
2A		6A		AA		EA	(LONG.DELAY)
2B	(MESSAGE)	6B		AB	(MEM/ASCII)	EB	
2C		6C		AC		EC	
2D		6D		AD		ED	
2E		6E		AE		EE	
2F		6F		AF		EF	
30		70		B0		F0	
31		71		B1		F1	
32		72		B2		F2	
33		73		B3		F3	(LAMP.TEST)
34		74		B4		F4	
35		75		B5		F5	
36		76		B6		F6	
37		77		B7		F7	
38		78		B8		F8	
39	(BILLBOARD)	79		B9		F9	
3A		7A		BA		FA	
3B		7B		BB		FB	
3C		7C		BC		FC	
3D		7D		BD		FD	
3E		7E		BE		FE	
3F		7F		BF		FF	

Figure 4-4. 256-Byte Memory Map—7303 Alphanumeric Display Subroutines.

PAGE ADDRESS 11

LINE	LABEL	LINE	LABEL	LINE	LABEL	LINE	LABEL
00	DISPLAY.DEMO	40		80		C0	CALCULATOR
01		41		81		C1	
02		42		82		C2	
03		43		83		C3	
04		44		84		C4	
05		45		85		C5	
06		46		86		C6	
07		47		87		C7	
08		48		88		C8	
09		49		89		C9	
0A		4A		8A		CA	
0B		4B		8B		CB	
0C		4C		8C		CC	
0D		4D		8D		CD	
0E		4E		8E		CE	
0F		4F		8F		CF	
10		50		90		D0	
11		51		91		D1	
12		52		92		D2	
13		53		93		D3	
14		54		94		D4	
15		55	(READ.KEY)	95		D5	
16		56		96		D6	
17		57		97		D7	
18		58		98		D8	
19		59		99		D9	
1A		5A		9A		DA	
1B	DISPLAY.SELF	5B		9B		DB	
1C		5C		9C		DC	
1D		5D		9D		DD	
1E		5E		9E		DE	
1F		5F		9F		DF	
20		60		A0		E0	
21		61		A1		E1	
22		62		A2		E2	
23		63		A3		E3	
24		64	(DECODE.KEY)	A4		E4	
25		65		A5		E5	
26		66		A6		E6	
27		67		A7	(SCAN)	E7	
28		68		A8		E8	
29		69		A9		E9	
2A		6A		AA		EA	
2B		6B		AB		EB	
2C		6C		AC		EC	
2D		6D		AD		ED	
2E		6E		AE		EE	
2F		6F		AF		EF	
30		70		B0		F0	KEY.TEST
31		71		B1		F1	
32		72		B2		F2	
33		73		B3		F3	
34		74		B4	(ROCKER.STATUS)	F4	
35		75		B5		F5	
36		76		B6		F6	
37		77		B7		F7	
38		78		B8		F8	
39		79		B9		F9	
3A		7A		BA		FA	
3B		7B		BB		FB	
3C		7C		BC		FC	
3D		7D		BD		FD	
3E		7E		BE		FE	
3F		7F		BF		FF	

Figure 4-5. 256-Byte Memory Map—7303 Keyboard Subroutines and Demonstration Programs.

PAGE ADDRESS 21 - RAM

LINE	LABEL	LINE	LABEL	LINE	LABEL	LINE	LABEL
00	TEXT START ADDR.	40		80		C0	
01		41		81		C1	
02	CHARACTER 7	42		82		C2	
03		43		83		C3	
04		44		84		C4	
05	MESSAGE	45		85		C5	
06	BUFFER	46		86		C6	
07		47		87		C7	
08		48		88		C8	
09	CHARACTER 0	49		89		C9	
0A		4A		8A		CA	
0B		4B		8B		CB	
0C		4C		8C		CC	
0D		4D		8D		CD	
0E		4E		8E		CE	
0F		4F		8F		CF	
10		50		90		D0	
11		51		91		D1	
12		52		92		D2	
13		53		93		D3	
14		54		94		D4	
15		55		95		D5	
16		56		96		D6	
17		57		97		D7	
18		58		98		D8	
19		59		99		D9	
1A		5A		9A		DA	
1B		5B		9B		DB	
1C		5C		9C		DC	
1D		5D		9D		DD	
1E		5E		9E		DE	
1F		5F		9F		DF	
20		60		A0		E0	
21		61		A1		E1	
22		62		A2		E2	
23		63		A3		E3	
24		64		A4		E4	
25		65		A5		E5	
26		66		A6		E6	
27		67		A7		E7	
28		68		A8		E8	
29		69		A9		E9	
2A		6A		AA		EA	
2B		6B		AB		EB	
2C		6C		AC		EC	
2D		6D		AD		ED	
2E		6E		AE		EE	
2F		6F		AF		EF	
30		70		B0		F0	
31		71		B1		F1	
32		72		B2		F2	
33		73		B3		F3	
34		74		B4		F4	
35		75		B5		F5	
36		76		B6		F6	
37		77		B7		F7	
38		78		B8		F8	
39		79		B9		F9	
3A		7A		BA		FA	
3B		7B		BB		FB	
3C		7C		BC		FC	
3D		7D		BD		FD	
3E		7E		BE		FE	
3F		7F		BF		FF	

NOTE

Only RAM locations 2100-2109 are used by the 7303; however, other Pro-Log software packages may use other portions of the processor card's onboard RAM memory. The designer should consult the users' manuals for the other cards being used to find the total amount of RAM needed for subroutine support.

Figure 4-6. 256-Byte Memory Map—7303 RAM "MAILBOX" Allocation.

ASCII Display Driver Module

This program module displays single ASCII characters at addressable positions in the 7303's alphanumeric display. The module-s subroutines handle all of the hardware requirements of the display; data communication with the subroutines is through "mailbox" locations in registers and memory. See Fig. 4-7 for flowchart.

This module consists of hardware-level subroutines that are used by other portions of the software package to create more complex display operations. The designer can use these subroutines to adapt the ASCII display to any desired format.

The subroutines are based on the 7303 programming requirements as shown in Section 3 of this manual.

- Displays ASCII characters shown in Fig. 3-3.
- Addressable display positions.
- Does all hardware manipulation.
- Not for cursor control—see cursor control module.
- See DISPLAY.DEMO program for application example.
- Contents:
  (DISPLAY)—Displays any one ASCII character in any one position
  (MEM.DISP)—Displays one ASCII character from memory
  (STROBE)—Pulses the display's WRITE line

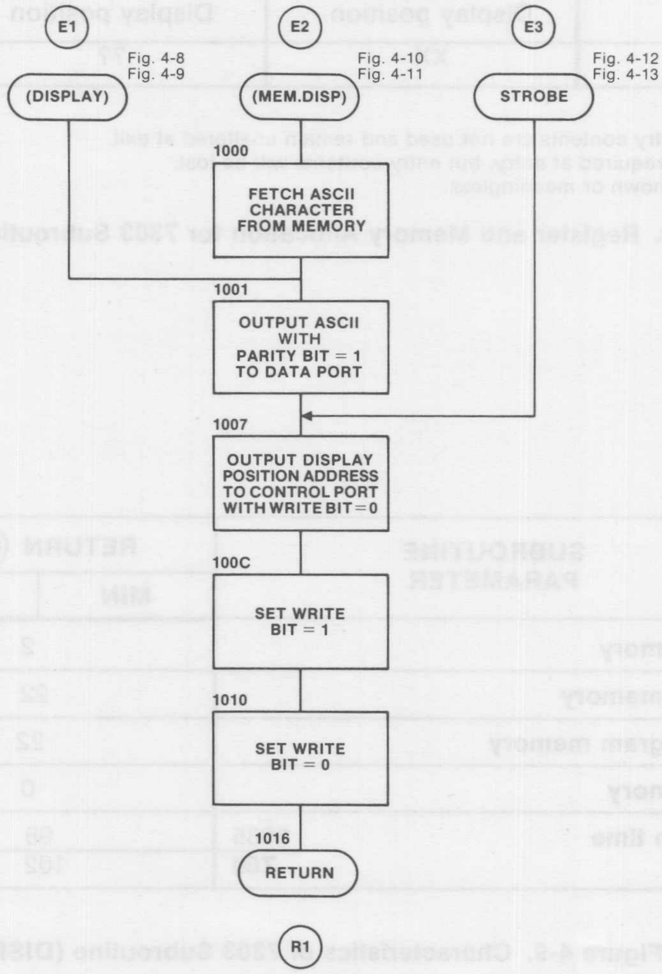


Figure 4-7. Flowchart—ASCII Display Driver Module for the 7303.

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This subroutine allows any one ASCII character to be displayed in any one of the eight alphanumeric display positions.

Preset register B with the desired display position. Use the 3-bit codes shown in Fig. 3-6 to specify one of eight positions, loading the code in register B's bits 2, 1, 0 with bits 3 through 7 = 0. For example, load register B with hexadecimal 06 to specify display position 6 (second display from the left).

Preset the accumulator (register A) with the desired ASCII character's hexadecimal code as shown in Fig. 3-3. The (DISPLAY) subroutine sets the parity bit (bit 7 = 1) as required by the 7303's displays, so that the character may be brought in from an external interface and displayed without code alteration.

Upon exit from the subroutine, the display position remains unaltered in register B, but the ASCII character in register A is lost.

PARAMETER ELEMENT	PARAMETER ADDRESS	ENTRY (E1)	RETURN (R1)	COMMENT
Register	A	ASCII character	??	—
Register	B	Display position	Display position	Load bits 0, 1, 2 only; bits 3 through 7 = 0
Register	F	XX	??	—

NOTES
1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-8. Register and Memory Allocation for 7303 Subroutine (DISPLAY).

SYMBOL	SUBROUTINE PARAMETER		RETURN (R1) MIN	RETURN (R1) MAX	UNITS
Ns	**Stack memory**		2		Bytes
Np	**Program memory**		22		Bytes
Npt	**Total program memory**		22		Bytes
Nr	**RAM memory**		0		Bytes
Te	**Execution time**	8085	98		Time states
		Z80	102		

Figure 4-9. Characteristics of 7303 Subroutine (DISPLAY).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This subroutine allows any one ASCII character to be read from memory and displayed in any one of the eight alphanumeric display positions.

Preset register B with the desired display position. Use the 3-bit codes shown in Fig. 3-6 to specify one of eight positions, loading the code in register B's bits 2, 1, 0 with bits 3 through 7 = 0. For example, load register B with hexadecimal 06 to specify display position 6 (second display from the left).

Register pair H,L is used as a memory pointer and must be preset to the address of the memory location in ROM or RAM, where the ASCII character to be displayed is located. Figure 3-3 shows the ASCII character set that can be displayed, and the range of codes that must be preloaded in memory before (MEM.DISP) can be used successfully. Use the (MEM/ASCII) subroutine in advance to translate raw binary memory data into ASCII if necessary.

Upon exit from the subroutine, the display position in register B and the memory address in pair H, L remain unaltered.

PARAMETER		ENTRY E2	RETURN R1	COMMENT
ELEMENT	**ADDRESS**			
Register	**H, L**	Memory address	Unaltered	H,L points to ASCII character in memory
Register	**B**	Display position	Unaltered	—
Register	**A**	XX	??	—
Register	**F**	XX	??	—

NOTES
1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-10. Register and Memory Allocation for 7303 Subroutine (MEM.DISP).

SYMBOL	SUBROUTINE PARAMETER		RETURN R1		UNITS
			MIN	**MAX**	
Ns	**Stack memory**		2		Bytes
Np	**Program memory**		23		Bytes
Npt	**Total program memory**		23		Bytes
Nr	**RAM memory**		1		Bytes
Te	**Execution time**	**8085**	105		Time states
		Z80	109		

Figure 4-11. Characteristics of 7303 Subroutine (MEM.DISP).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This captive subroutine is used by other subroutines to drive the 7303 display's write line (WR*) low/high/low, while maintaining the desired display-position-address constant. This is explained in detail in Section 3.

Use (STROBE) to adapt the 7303's display to an application for which the other subroutines in the software package are not suitable. It is important to note that other methods for driving the WR* control line may result in unwanted changes in the display, unless the programming rules outlined in Section 3 are followed.

PARAMETER		ENTRY E3	RETURN R1	COMMENT
ELEMENT	ADDRESS			
Register	A	Display position 0-7	??	—
Register	F	XX	??	—

NOTES
1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-12. Register and Memory Allocation for 7303 Subroutine (STROBE).

SYMBOL	SUBROUTINE PARAMETER		RETURN R1 MIN	RETURN R1 MAX	UNITS
Ns	Stack memory		2		Bytes
Np	Program memory		16		Bytes
Npt	Total program memory		16		Bytes
Nr	RAM memory		0		Bytes
Te	Execution time	8085	73		Time states
		Z80	76		

Figure 4-13. Characteristics of 7303 Subroutine (STROBE).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

Cursor Control Module

This program module controls the on/off state of the cursor characters. The module's subroutines handle all of the hardware requirements of the cursors. The full 8-position cursor on/off pattern is specified by a single 8-bit pattern preset in a register "mailbox." See Fig. 4-14 for flowchart.

The subroutines are based on the 7303 programming requirements as shown in Section 3 of this manual.

- Controls all eight cursor on/off states.
- Does all hardware manipulation.
- One 8-bit word specifies cursor pattern.
- See DISPLAY.DEMO program for application example.
- Contents:
  (CURSORS)—Turns on/off any combination of cursors.
  (CLR.CURSORS)—Removes cursors (not ASCII characters).

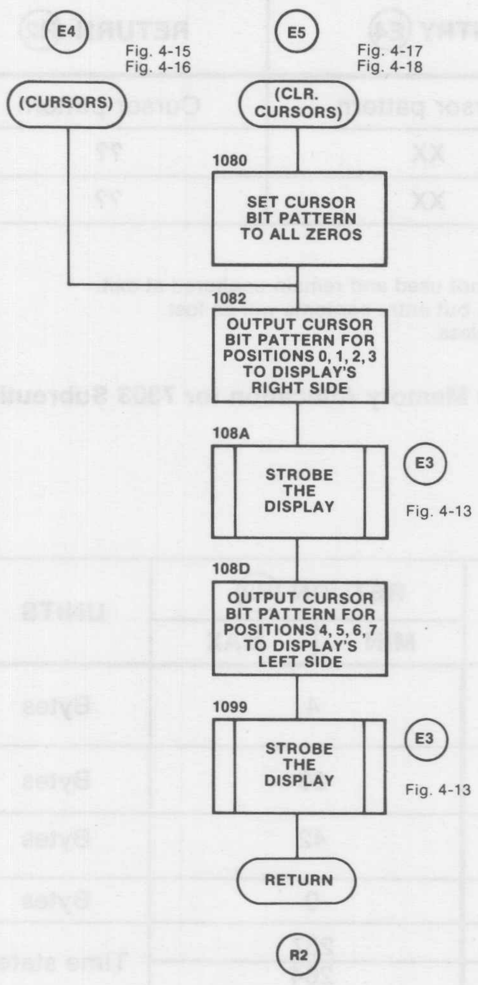


Figure 4-14. Flowchart—Cursor Control Module for the 7303.

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This subroutine allows any combination of cursors to be displayed or removed in one operation, using a single 8-bit word to specify the cursor on/off pattern.

> NOTE
> Each display position must have a valid ASCII character present in its character memory before it can display the cursor character. The SPACE character satisfies this requirement; use the (CLEAR. DISPLAY) or other subroutine to preload valid ASCII characters at least once, before using the (CURSORS) subroutine.

Preset register B with the desired cursor pattern. Register B's bits have 1:1 correspondence with the eight displays (bit 7 controls the cursor in display position 7). Set the bit = 1 to turn the cursor on, or bit = 0 to remove the cursor. Upon exit, the cursor pattern in register B is unaltered.

PARAMETER		ENTRY E4	RETURN R2	COMMENT
ELEMENT	ADDRESS			
Register	B	Cursor pattern	Cursor pattern	—
Register	A	XX	??	—
Register	F	XX	??	—

NOTES
1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-15. Register and Memory Allocation for 7303 Subroutine (CURSORS).

SYMBOL	SUBROUTINE PARAMETER		RETURN R2 MIN	MAX	UNITS	COMMENT
Ns	Stack memory		4		Bytes	Subroutine (STROBE) used. E3 Fig. 4-12.
Np	Program memory		26		Bytes	—
Npt	Total program memory		42		Bytes	—
Nr	RAM memory		0		Bytes	—
Te	Execution time	8085	257		Time states	Absolute time varies
		Z80	264			

Figure 4-16. Characteristics of 7303 Subroutine (CURSORS).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This subroutine removes all eight cursors from the alphanumeric display. The ASCII characters loaded into the display's ASCII memories before displaying the cursors will reappear when the cursors are removed.

Register B is cleared by this subroutine.

PARAMETER		ENTRY E5	RETURN R2	COMMENT
ELEMENT	ADDRESS			
Register	B	XX	00	—
Register	A	XX	??	—
Register	F	XX	??	—

NOTES
1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-17. Register and Memory Allocation for 7303 Subroutine (CLR.CURSORS).

SYMBOL	SUBROUTINE PARAMETER		RETURN R2		UNITS	COMMENT
			MIN	MAX		
Ns	Stack memory		4		Bytes	Subroutine (STROBE) used. E3 Fig. 4-12.
Np	Program memory		28		Bytes	—
Npt	Total program memory		44		Bytes	—
Nr	RAM memory		0		Bytes	—
Te	Execution time	8085	264		Time states	Absolute time varies
		Z80	271			

Figure 4-18. Characteristics of 7303 Subroutine (CLR.CURSORS).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

Display Service Routines Module

This program module provides hardware-level service routines for clearing and testing the 7303's alphanumeric display. See Fig. 4-19 for flowchart.

The subroutines in this module are used to initialize the 7303 after power-on, to clear the display when desired, and to provide general service functions needed in incoming inspection, field testing, and repair of the 7303 card.

- (CLEAR.DISPLAY) removes ASCII characters only.
- (CLEAR.BOTH) removes both ASCII and cursor characters.
- (DISPLAY.8) allows the testing of each ASCII character in each display; it finds bad latches, decoders, drivers, and LED segments.
- (LAMP.TEST) allows the testing of all alphanumeric and binary LED segments.
- See DISPLAY.TEST program for application example.

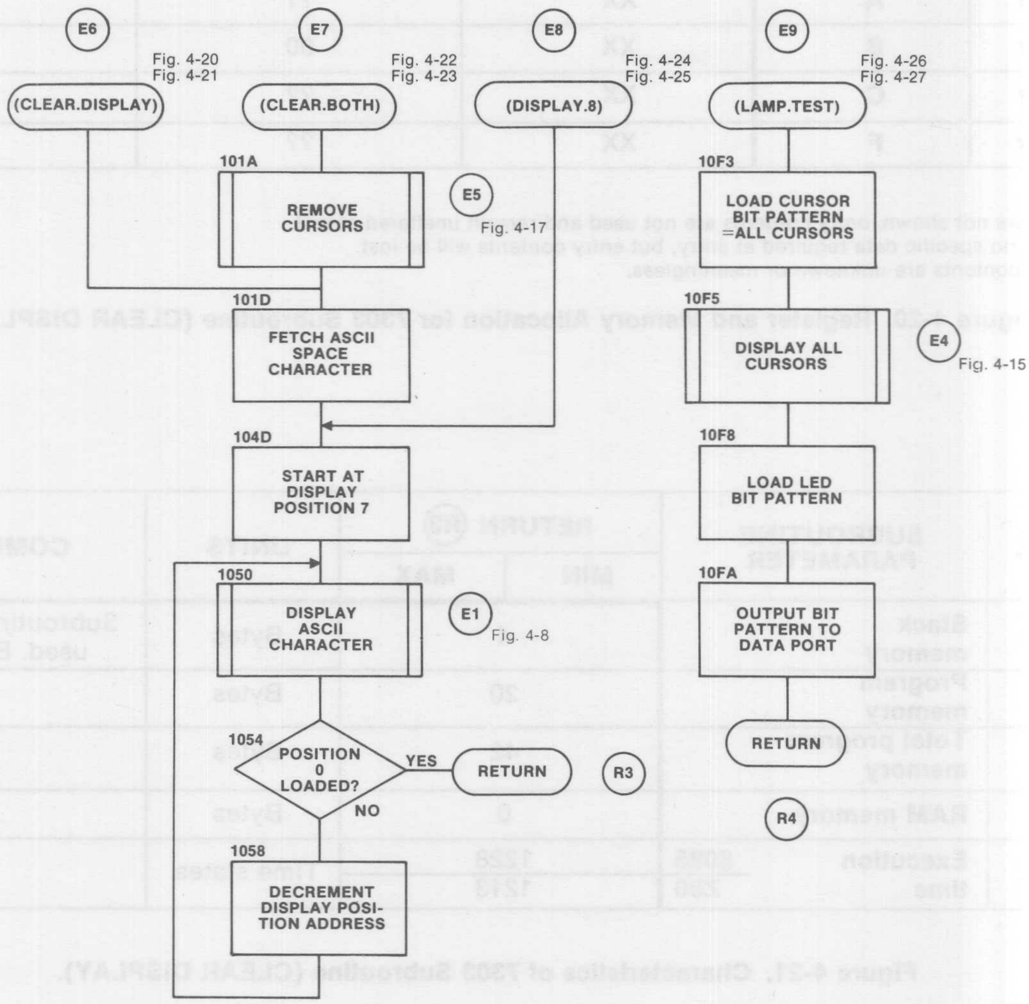


Figure 4-19. Flowchart—Display Service Module for the 7303.

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This subroutine blanks the alphanumeric display by loading the SPACE character in each of the eight positions.

Note that the cursors are unaltered by this subroutine. Use (CLR.CURSORS) to remove cursor characters. Register B is cleared by this subroutine.

PARAMETER		ENTRY (E6)	RETURN (R3)	COMMENT
ELEMENT	**ADDRESS**			
Register	**A**	XX	??	—
Register	**B**	XX	00	—
Register	**C**	XX	??	—
Register	**F**	XX	??	—

NOTES
1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-20. Register and Memory Allocation for 7303 Subroutine (CLEAR DISPLAY).

SYMBOL	SUBROUTINE PARAMETER		RETURN (R3)		UNITS	COMMENT
			MIN	**MAX**		
Ns	**Stack memory**		4		Bytes	Subroutine (DISPLAY) used. E1 Fig. 4-8.
Np	**Program memory**		20		Bytes	—
Npt	**Total program memory**		42		Bytes	—
Nr	**RAM memory**		0		Bytes	—
Te	**Execution time**	**8085**	1228		Time states	—
		Z80	1213			

Figure 4-21. Characteristics of 7303 Subroutine (CLEAR DISPLAY).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This subroutine removes all cursor characters from the display and blanks the alphanumeric display by loading the SPACE character in all eight positions.

Register B is cleared by this subroutine.

PARAMETER		ENTRY E7	RETURN R3	COMMENT
ELEMENT	ADDRESS			
Register	A	XX	??	—
Register	B	XX	00	—
Register	C	XX	??	—
Register	D	XX	??	—

NOTES

1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-22. Register and Memory Allocation for 7303 Subroutine (CLEAR.BOTH).

SYMBOL	SUBROUTINE PARAMETER		RETURN R3 MIN	RETURN R3 MAX	UNITS	COMMENT
Ns	Stack memory		4		Bytes	See note
Np	Program memory		23		Bytes	—
Npt	Total program memory		73		Bytes	—
Nr	RAM memory		0		Bytes	—
Te	Execution time	8085 Z80	1510 1501		Time states	—

NOTE

Subroutines used: (CLR.CURSORS) E5 Fig. 4-17.
(DISPLAY) E1 Fig. 4-8.

Figure 4-23. Characteristics of 7303 Subroutine (CLEAR.BOTH).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This subroutine displays the same ASCII character in all eight display positions simultaneously. It is a service routine for implementing the (CLEAR.DISPLAY) subroutine, and is useful for alphanumeric display test operations.

Preset the accumulator (register A) with the character to be displayed.

Upon exit, register C contains the ASCII character displayed and register B is cleared.

PARAMETER		ENTRY (E8)	RETURN (R3)	COMMENT
ELEMENT	ADDRESS			
Register	A	ASCII character	??	—
Register	B	XX	00	—
Register	C	XX	ASCII character	—
Register	F	XX	??	—

NOTES
1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-24. Register and Memory Allocation for 7303 Subroutine (DISPLAY.8).

SYMBOL	SUBROUTINE PARAMETER		RETURN (R3)		UNITS	COMMENT
			MIN	MAX		
Ns	Stack memory		4		Bytes	Subroutine (DISPLAY) used. E1 Fig. 4-8.
Np	Program memory		15		Bytes	—
Npt	Total program memory		37		Bytes	—
Nr	RAM memory		0		Bytes	—
Te	Execution time	8085	1211		Time states	—
		Z80	1196			

Figure 4-25. Characteristics of 7303 Subroutine (DISPLAY.8).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This subroutine displays the cursor character in all eight display positions (illuminating all LED segments in the alphanumeric display). It also writes hexadecimal FF to the 7303's output data port, which illuminates all of the eight binary LEDs located directly below the alphanumeric display.

NOTE

All the keyboard and display routines, except (ROCKER.STATUS) and (LAMP.TEST), will write to the output data port, altering the all-on state of the binary LED display.

Consequently the designer should follow (LAMP.TEST) with a time delay, or other method, that gives the operator an opportunity to examine the LED display before executing other portions of the software package.

PARAMETER		ENTRY E9	RETURN R4	COMMENT
ELEMENT	ADDRESS			
Register	A	XX	??	—
Register	B	XX	??	—
Register	F	XX	??	—

NOTES
1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-26. Register and Memory Allocation for 7303 Subroutine (LAMP.TEST).

SYMBOL	SUBROUTINE PARAMETER		RETURN R4		UNITS	COMMENT
			MIN	MAX		
Ns	Stack memory		4		Bytes	See Note
Np	Program memory		11		Bytes	—
Npt	Total program memory		53		Bytes	—
Nr	RAM memory		0		Bytes	—
Te	Execution time	8085	313		Time states	—
		Z80	320			

Figure 4-27. Characteristics of 7303 Subroutine (LAMP.TEST).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

Hexadecimal/ASCII Conversion Module

This program module converts binary data, in registers and in blocks of memory, into ASCII-encoded data suitable for display by the 7303 and for transmission via RS-232, TTY, and other media. See Fig. 4-28 for flowchart.

- Accepts one 4-bit hexadecimal digit (0000 through 1111 binary or 0-F hexadecimal) from a register and outputs one 8-bit ASCII character, 0-9 or A-F.
- Accepts two 4-bit hexadecimal digits in each of 1-256 locations anywhere in memory and outputs 2-512 ASCII characters to RAM memory.
- Produces ASCII characters 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and A, B, C, D, E, F, (upper case only) with parity bit set (bit 7 = 1).
- See DISPLAY.SELF and KEY.TEST for application examples.
- Contents:
  (HEX/ASCII)—Converts one hexadecimal digit to one ASCII character code.
  (MEM/ASCII)—Converts block of binary in memory into displayable ASCII codes.
  (DISP.HEX)—Combines (HEX/ASCII) and (DISPLAY).
  (DISP.2.IN.C)—Displays two hexadecimal digits in internal register.

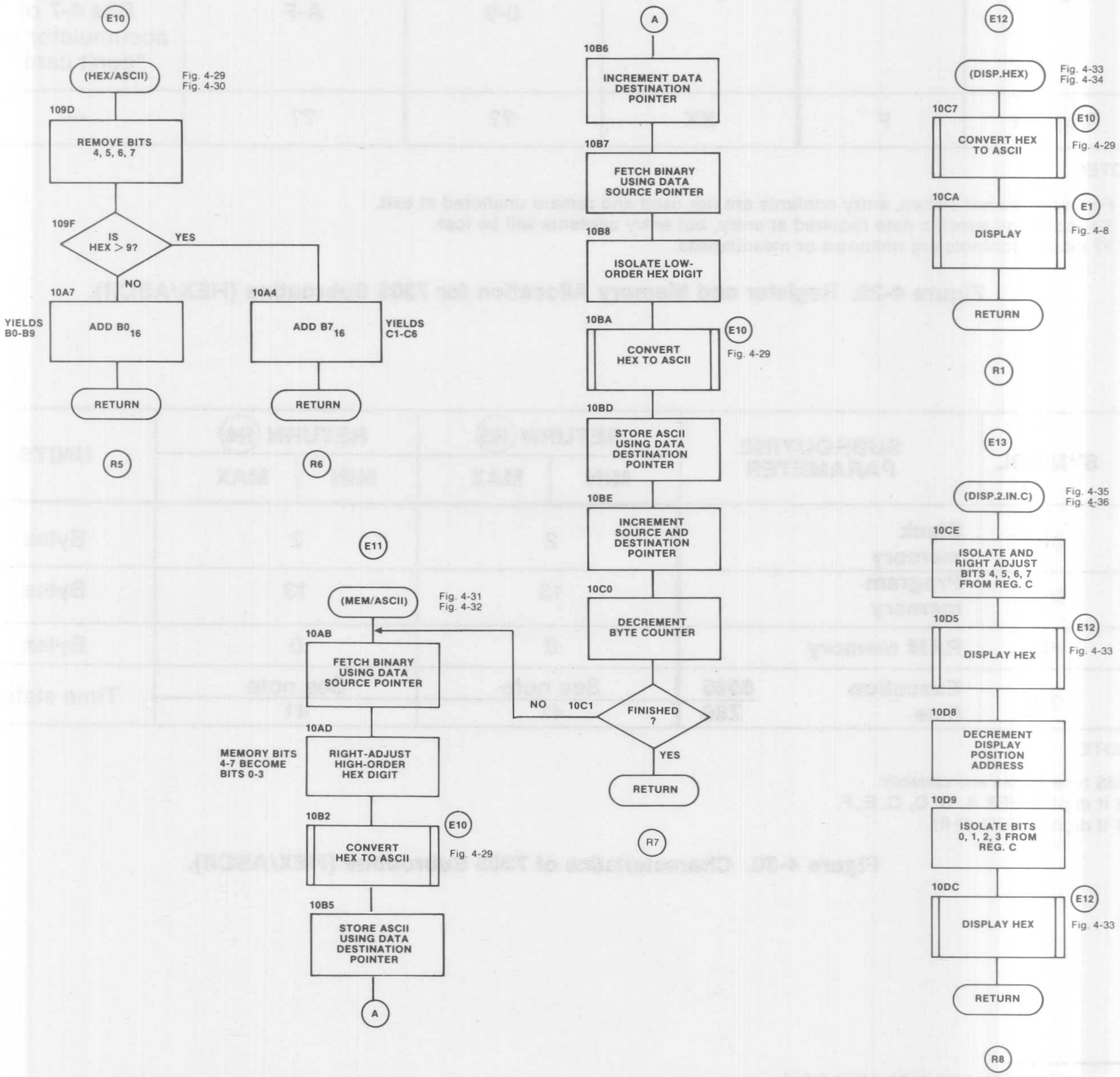


Figure 4-28. Flowchart—Hexadecimal/ASCII Conversion Module for the 7303.

( ) Denotes subroutine label; * Low level active; E/R Entry/return path identifier encircled.

This subroutine converts a 4-bit binary/hexadecimal code into one of 16 ASCII characters: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, or A, B, C, D, E, F (upper case only) with parity set (bit 7 = 1).

The ASCII codes returned by the subroutine for the 16 characters are shown in Fig. 3-3.

Enter with the hexadecimal digit loaded in bits 3, 2, 1, 0 of the accumulator (register A). The most significant bits (4 through 7) of register A are "don't care" and will be masked by the subroutine.

Upon exit, the ASCII character code is stored in register A, bits 7 through 0, and the input binary code is lost.

PARAMETER		ENTRY E10	RETURN R5	RETURN R6	COMMENT
ELEMENT	ADDRESS				
Register	A	Hexadecimal digit 0-F	ASCII character 0-9	ASCII character A-F	Converts bits 0-3. Bits 4-7 of accumulator are "don't care."
Register	F	XX	??	??	—

NOTES

1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-29. Register and Memory Allocation for 7303 Subroutine (HEX/ASCII).

SYMBOL	SUBROUTINE PARAMETER		RETURN R5 MIN	RETURN R5 MAX	RETURN R6 MIN	RETURN R6 MAX	UNITS
Ns	Stack memory		2		2		Bytes
Np	Program memory		13		13		Bytes
Nr	RAM memory		0		0		Bytes
Te	Execution time	8085	See note		See note		Time states
		Z80	41		41		

NOTE

8085 time states are variable:
38 if digit is HEX A, B, D, C, E, F.
41 if digit is BCD (0-9).

Figure 4-30. Characteristics of 7303 Subroutine (HEX/ASCII).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This subroutine converts a block of memory locations (each containing 8-bit binary data expressed as two 4-bit hexadecimal digits) into a block of data with one ASCII character in each location.

> NOTE
>
> Since each 4-bit half of the binary input data is converted into one 8-bit ASCII character, the resulting block of output data written to RAM by this subroutine is twice as large as the block of input binary data.

Preset register pair H,L with the first (lowest) address in the block of input binary data in memory, which may be in ROM or RAM space.

Preset register pair D,E with the first (lowest) address in the block of output ASCII character data in memory, which can be in RAM only.

Preset register B with the number of bytes in the block of input binary data. Use 01 for one byte, 02 for two bytes, etc., FF for 255 bytes, and 00 for 256 bytes.

Upon exit, register B is cleared; register pair H,L points at the next location past the input data block; register pair D,E points at the next location past the output data block.

PARAMETER		ENTRY E11	RETURN R7	COMMENT
ELEMENT	**ADDRESS**			
Register	**H,L**	MEM pointer input	Last input +1	Note 4
Register	**D,E**	MEM pointer output	Last input +1	—
Register	**B**	Input data counter	00	—

NOTES

1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.
4. First byte in ASCII output memory block is ASCII conversion of bits 4-7 of first byte in binary block; second byte in ASCII output memory block is bits 0-3 of first byte in binary block, etc.

Figure 4-31. Register and Memory Allocation for 7303 Subroutine (MEM/ASCII).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

Subroutine: (MEM/ASCII)

SYMBOL	SUBROUTINE PARAMETER		RETURN (R7) MIN	MAX	UNITS	COMMENT
Ns	Stack memory		4		Bytes	Subroutine (HEX/ASCII) used. E10 Fig. 4-29.
Np	Program memory		26		Bytes	—
Npt	Total program memory		39		Bytes	—
Nr	RAM memory		3	768	Bytes	—
Te	Execution time	8085	Note 1		Time states	—
		Z80	Note 2			

NOTES

1. 8085 time states are variable; first binary memory location converted:
   229 if both digits are HEX A, B, C, D, E, F
   226 if one digit is BCD (0-9)
   223 if both digits are BCD.

Each additional location: subtract 7 time states from above totals.

2. Z80 time states:
   —First binary memory location converted, 230 time states.
   —Each additional binary location, 220 time states.

Figure 4-32. Characteristics of 7303 Subroutine (MEM/ASCII).

This subroutine uses lower-level subroutines to display one hexadecimal digit (0-9 or A-F) in any one of the eight display positions.

Preset register B with the desired display position. Use the 3-bit codes shown in Fig. 3-6 to specify one of eight positions, loading the code in register B's bits 2, 1, 0 with bits 3 through 7 = 0. For example, load register B with hexadecimal 06 to specify display position 6 (second display from the left).

Preset the accumulator (register A) with the binary bit pattern of the hexadecimal digit (0000 through 1111 binary) in register A's bits 3, 2, 1, 0; bits 4 through 7 are "don't care" and may contain any bit pattern.

Upon exit, the display position remains unaltered in register B, but the hexadecimal digit in the accumulator is lost.

PARAMETER		ENTRY E12	RETURN R1	COMMENT
ELEMENT	**ADDRESS**			
Register	**A**	Hexadecimal digit 0-F	??	
Register	**F**	XX	??	—
Register	**B**	Display position 0-7	Display position 0-7	—

NOTES

1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-33. Register and Memory Allocation for 7303 Subroutine (DISP.HEX).

SYMBOL	SUBROUTINE PARAMETER		RETURN R1 MIN	RETURN R1 MAX	UNITS	COMMENT
Ns	**Stack Memory**		4		Bytes	Subroutine (HEX/ASCII) used. E10 Fig. 4-29.
Np	**Program memory**		6		Bytes	—
Npt	**Total program memory**		41		Bytes	—
Nr	**RAM memory**		0		Bytes	—
Te	**Execution time**	**8085**	See note		Time states	—
		Z80	180			

NOTE

8085 time states depend on data:
177 if digit is HEX A, B, C, D, E, F
174 if digit is HEX (0-9).

Figure 4-34. Characteristics of 7303 Subroutine (DISP.HEX).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This subroutine converts the two 4-bit hexadecimal digits in register C into two 8-bit ASCII characters in the range 0-9 or A-F, and it displays them in two adjacent display positions.

Preset register C with the data to be displayed. The subroutine converts register C's bits 4-7 into an ASCII character and displays the character in the display position specified below. Then, register C's bits 0-3 are converted into a second ASCII character and displayed in the display position immediately to the right of the position specified.

Preset register B with the leftmost of two desired display positions. Use the 3-bit codes shown in Fig. 3-6 to specify one of **seven** positions, loading the code in register B's bits 2, 1, 0 with bits 3 through 7 = 0.

CAUTION

Do not specify position zero; register B should contain the combinations 01, 02, 03, 04, 05, 06, or 07 only after this step.

Upon exit, register B will have been decremented by 1 from its initial condition, and two hexadecimal digits in register C will have been unaltered.

PARAMETER		ENTRY E13	RETURN R8	COMMENT
ELEMENT	ADDRESS			
Register	C	Two hex digits	Two hex digits	Note 4
Register	B	Display position N	Position N-1	Note 5
Register	F	XX	??	

NOTES

1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.
4. Bits 4-7 converted to an ASCII code and displayed in specified display position: bits 0-3 displayed as ASCII character in adjacent display position on right.
5. Bits 0-2 specify display position and **must be nonzero** (do not specify position 0).

Figure 4-35. Register and Memory Allocation for 7303 Subroutine (DISP.2.IN.C).

SYMBOL	SUBROUTINE PARAMETER		RETURN R8 MIN	MAX	UNITS	COMMENT
Ns	Stack memory		6		Bytes	Note 2
Np	Program memory		17		Bytes	—
Npt	Total program memory		58		Bytes	—
Nr	RAM memory		0		Bytes	
Te	Execution time	8085 Z80	See note 418		Time states	—

NOTES

1. 8085 time states are variable:
   413 if both digits are HEX A, B, C, D, E, F
   410 if one character is BCD (0-9)
   407 if both characters are BCD.
2. Subroutine used: (DISP.HEX) E12 Fig. 4-33.

Figure 4-36. Characteristics of 7303 Subroutine (DISP.2.IN.C).

( ) Denotes subroutine label; * Low level active; E/R Entry/return path identifier encircled.

Formatted Messages Module

This program module (see Fig. 4-37 for flowchart) uses the hardware-level subroutines to format and display messages of the designer's choice. Two styles are available:

1. (MESSAGE) displays a static 8-character ASCII message from anywhere in memory.
2. (BILLBOARD) displays a dynamic message of 8 characters or more from anywhere in memory, rotated across the display in billboard fashion.

These formats can be used repeatedly and in combination to show system status, prompt the system's operator, and other applications. Use the Hexadecimal/ASCII Conversion Module and the Cursor Control Module for more variations on these basic formats.

- Static 8-character ASCII message display.
- Dynamic "billboard" display for messages of 8 characters or longer.
- See DISPLAY.DEMO and CALCULATOR demonstration programs for application examples.

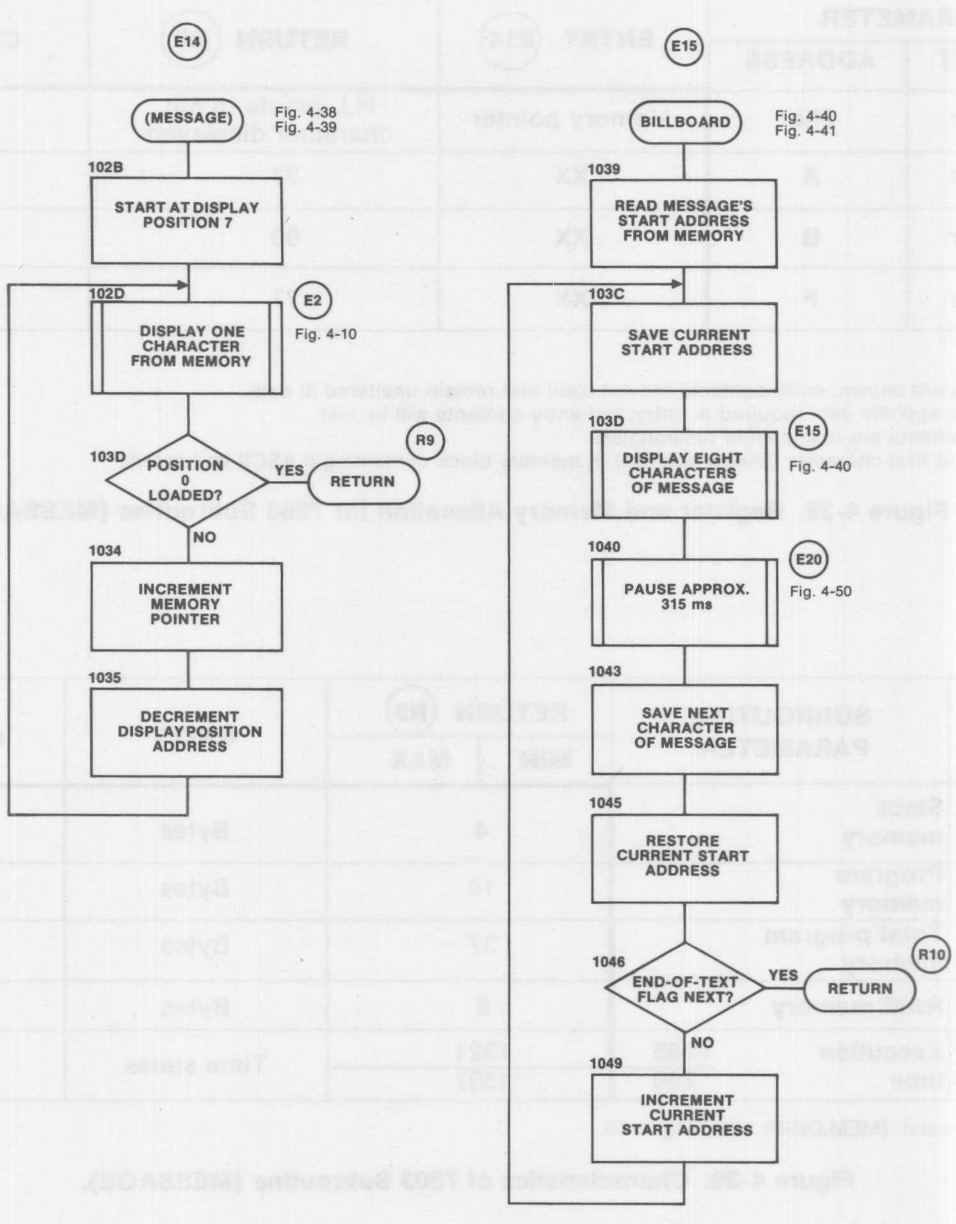


Figure 4-37. Flowchart—Formatted Messages Module for the 7303.

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This subroutine displays an 8-character ASCII message from anywhere in memory. The job of creating a variable-format display can be simplified by manipulating an 8-character text buffer in RAM memory, then unconditionally displaying the content of the buffer using (MESSAGE) after each alteration of the buffer's content. See (BILLBOARD) for an example.

Preset register pair H,L with the first (lowest address) memory location to be displayed. This can be in either RAM or ROM memory, and it appears in the leftmost display position. The (MESSAGE) subroutine fills the display from left to right, incrementing the H,L register pair each time until all eight locations are loaded with ASCII characters.

Upon exit, register B is cleared and register pair H,L points at the last character (highest memory address) displayed. The displayed memory locations are unaltered.

PARAMETER		ENTRY (E14)	RETURN (R9)	COMMENT
ELEMENT	ADDRESS			
Register	H,L	Memory pointer	H,L points to 8th character displayed	Note 4.
Register	A	XX	??	—
Register	B	XX	00	—
Register	F	XX	??	—

NOTES
1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.
4. Set to point at first character (lowest address) in memory block containing 8 ASCII characters.

Figure 4-38. Register and Memory Allocation for 7303 Subroutine (MESSAGE).

SYMBOL	SUBROUTINE PARAMETER		RETURN (R9) MIN	MAX	UNITS	COMMENTS
Ns	Stack memory		4		Bytes	See note
Np	Program memory		14		Bytes	—
Npt	Total program memory		37		Bytes	—
Nr	RAM memory		8		Bytes	—
Te	Execution time	8085	1321		Time states	—
		Z80	1337			

Note: Subroutine used: (MEM.DISP) E2 Fig. 4-10.

Figure 4-39. Characteristics of 7303 Subroutine (MESSAGE).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This subroutine shifts a long message of eight or more ASCII characters across the 8-position display, leaving each character combination in the display for about 300 ms before shifting. The text can begin anywhere in memory and be of any desired length.

The entire message is displayed once, then the routine exits with the last eight characters in the message remaining in the display. This gives the program an opportunity to alter the message before the next iteration, if the text is loaded in RAM. Execute the (BILLBOARD) subroutine repeatedly to create an endlessly rotating billboard effect.

The message consists of any number of ASCII characters (limited by the size of the user's contiguous memory), terminated by hexadecimal FF. The subroutine will exit after the FF code is encountered.

Preset two sequential memory locations labeled TEXT.START (See RAM map, Fig. 4-6) with the first (lowest) memory address of the ASCII message. Do this only once—the subroutine can then be used repeatedly without additional presets.

Upon exit, the text-start address in RAM is unaltered.

NOTE

When constructing the message, we recommend that the SPACE character (hexadecimal A0) be loaded as the **first seven** and **last eight characters** in the text. This produces the smooth transition from the end of the message to the beginning, which is characteristic of billboards.

PARAMETER		ENTRY E15	RETURN R10	COMMENT
ELEMENT	**ADDRESS**			
Register	**H,L**	Memory pointer	Memory points	—
Register	**A,F**	XX	??	—
Register	**B**	XX	??	—
Register	**C**	XX	??	—

NOTES
1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-40. Register and Memory Allocation for 7303 Subroutine (BILLBOARD).

SYMBOL	SUBROUTINE PARAMETER		RETURN R10 MIN	MAX	UNITS	COMMENTS
Ns	**Stack memory**		6		Bytes	Note 3
Np	**Program memory**		20		Bytes	—
Npt	**Total program memory**		76		Bytes	—
Nr	**RAM memory**		10	User dependent	Bytes	—
Te	**Execution time**	**8085**		Note 1	Time states	—
		Z80		Note 2		

NOTES
1. 8085—First 8 characters: 1,433 time states + delay. Each additional character = 1,427 time states + delay.
2. Z80—First 8 characters: 1,920 time states + delay. Each additional character = 1,919 time states + delay.
3. Subroutines used: (MESSAGE) E14 Fig. 4-38. (DISPLAY.DELAY) E20 Fig. 4-50.

Figure 4-41. Characteristics of 7303 Subroutine (BILLBOARD).

( ) Denotes subroutine label * Low level active E/R Entry/return path identifier encircled

Key and Switch Data Entry Module

This module controls the 7303's hexadecimal keyboard and translates the general purpose rocker-switch states into program status information for decision making. See Fig. 4-42 for the flowchart.

The module contains subroutines that perform all procedural requirements of keyboard-reading and decoding as well as general subroutines that allow the user to design a special keyboard procedure.

- Returns a unique 5-bit hexadecimal number (range 00-17) for each of 24 uncommitted keys—use table lookup and change key labels to perform any numeric or nonnumeric program function.
- Performs all procedures including switch debounce, noise rejection; activate on depression/ignore key release.
- (READ.KEY) reads single key only—use other subroutines in module for multiple key closures and different procedures.
- (ROCKER.STATUS) moves on/off states of switches to processor status flags for conditional jumps.
- See CALCULATOR and KEY.TEST for application examples.

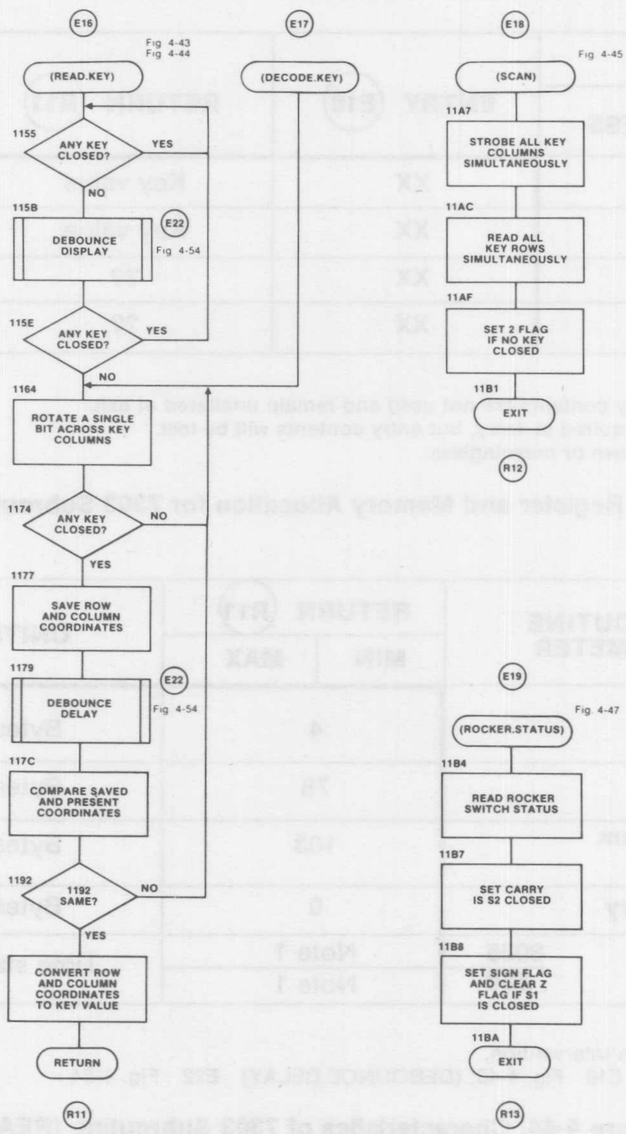


Figure 4-42. Flowchart—Key and Switch Data Entry Module for the 7303.

( )	Denotes subroutine label
*	Low level active
E/R	Entry/return path identifier encircled

This subroutine is recommended for most keyboard read/decode operations, regardless of the functional assignments associated with the keys and their labels.

(READ.KEY) begins by determining that the keyboard is idle, and it will not proceed until it is. It then waits until a key is pressed, and it decodes the key's value after rejecting noise and switch bounce. Once entered, the subroutine cannot exit until a valid key closure has occurred.

The 7303 card is shipped with labels attached to the keys. The hexadecimal labels in the 00 to 17 range are the values that will be decoded by (READ.KEY) for the key pressed (Fig. 3-1 shows label values; note that the RESET key is electrically isolated from the other 24 keys and is not read by this subroutine). Even if you relabel the keys to nonnumeric functions (such as MOTOR START or CLEAR ENTRY), you would still use this subroutine to read the keyboard. Simply use the decoded value to determine which function to perform—see the CALCULATOR demonstration program for an example.

Upon exit, the decoded key value is in both register A (for immediate use) and in register B, where it can be held momentarily if the accumulator is needed for other functions.

PARAMETER		ENTRY E16	RETURN R11	COMMENT
ELEMENT	ADDRESS			
Register	A	XX	Key value	—
Register	B	XX	Key value	—
Register	B	XX	??	—
Register	F	XX	??	—

NOTES
1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-43. Register and Memory Allocation for 7303 Subroutine (READ.KEY).

SYMBOL	SUBROUTINE PARAMETER		RETURN R11		UNITS	COMMENTS
			MIN	MAX		
Ns	Stack memory		4		Bytes	Note 2
Np	Program memory		78		Bytes	—
Npt	Total program memory		103		Bytes	—
Nr	RAM memory		0		Bytes	—
Te	Execution time	8085	Note 1		Time states	—
		Z80	Note 1			

NOTES
1. Not predictable, due to human intervention.
2. Subroutines used: (SCAN) E18 Fig. 4-45; (DEBOUNCE.DELAY) E22 Fig. 4-54

Figure 4-44. Characteristics of 7303 Subroutine (READ.KEY).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This subroutine is similar to (READ.KEY), but it omits testing for the keyboard-idle condition before reading and decoding the key.

> IMPORTANT
>
> Unless the user adds additional procedural instructions when using the (DECODE.KEY) subroutine, the system may react more than once to the same key closure, causing a system error.
>
> This subroutine is provided **only** to allow the user to design a special keyboard read/decode procedure, in applications where the (READ.KEY) subroutine, which is normally recommended, is not useful.

For decoded key values and their location upon exit, see (READ.KEY).

This subroutine is used to check keyboard status, by determining whether any key is closed. If a key closure is detected, (SCAN) is unable to determine which key is closed.

The subroutine is included for use with (DECODE.KEY), allowing the user to design a keyboard read/decode procedure if the (READ.KEY) subroutine, which is normally recommended, is not useful.

Upon exit, the Z (zero) flag can be tested to determine if any key has been pressed. The conditional jump instructions are as follows:

1. The **JP Z0** instruction will result in a jump if a key is closed; no jump will occur if all keys are idle.
2. The **JP Z1** instruction will result in a jump if all keys are idle; no jump will occur if any key is pressed.

Note that (SCAN) can be mislead by switch bounce or noise. For this reason, we recommend that no program decision be made until the (SWITCH.DEBOUNCE) subroutine has been executed and (SCAN) has been repeated. See the (READ.KEY) flowchart (Fig. 4-42) for an example.

PARAMETER		ENTRY E18	RETURN R12	COMMENT
ELEMENT	ADDRESS			
Register	A	XX	??	—
Register	F	XX	Keyboard status	—

NOTES
1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-45. Register and Memory Allocation for 7303 Subroutine (SCAN).

SYMBOL	SUBROUTINE PARAMETER		RETURN R12 MIN	RETURN R12 MAX	UNITS	COMMENTS
Ns	Stack memory		2		Bytes	—
Np	Program memory		11		Bytes	—
Npt	Total program memory		11		Bytes	—
Nr	RAM memory		0		Bytes	—
Te	Execution time	8085	52		Time states	—
		Z80	54			

Figure 4-46. Characteristics of 7303 Subroutine (SCAN).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This subroutine moves the on/off status of the two rocker switches into the processor's flag (register F). This allows conditional jump instructions to alter the program flow according to the on/off (closed/open) status of the two uncommitted rocker switches.

Upon exit, use the following conditional jump instructions:

1. **JP S1** or **JP Z0** will cause a jump if the right-hand rocker switch is closed.
2. **JP S0** or **JP Z1** will cause a jump if the right-hand rocker switch is open.
3. **JP C1** will cause a jump if the left-hand rocker switch is closed.
4. **JP C0** will cause a jump if the left-hand rocker switch is open.

PARAMETER		ENTRY E19	RETURN R13	COMMENT
ELEMENT	ADDRESS			
Register	A	XX	??	—
Register	F	XX	Switch status	—

NOTES
1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-47. Register and Memory Allocation for 7303 Subroutine (ROCKER.STATUS).

SYMBOL	SUBROUTINE PARAMETER		RETURN R13		UNITS	COMMENTS
			MIN	MAX		
Ns	Stack memory		2		Bytes	—
Np	Program memory		7		Bytes	—
Npt	Total program memory		7		Bytes	—
Nr	RAM memory		0		Bytes	—
Te	Execution time	8085	35		Time states	—
		Z80	36			

Figure 4-48. Characteristics of 7303 Subroutine (ROCKER.STATUS).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

Auxiliary Timing Module

This module contains captive subroutines used by the key and switch data entry module and by the demonstration and test programs in the 7303's software package. See Fig. 4-49 for flowchart.

The subroutines in this module are designed to provide satisfactory operation with a wide variety of microprocessor types, including 8080, 8085A, Z80, and NSC 800, all presumed to operate at the maximum clock frequency. Accordingly, these subroutines are not capable of generating accurate timing and should not be used in any application requiring accurate timing. They are intended only to reduce the processor's execution rate to maintain human readability of the display in the demonstration programs, and to provide switch debounce time for the 7303's keyboard.

- Approximate time delays used for display readability and switch debouncing.
- Captive subroutines used by other 7303 program modules only.

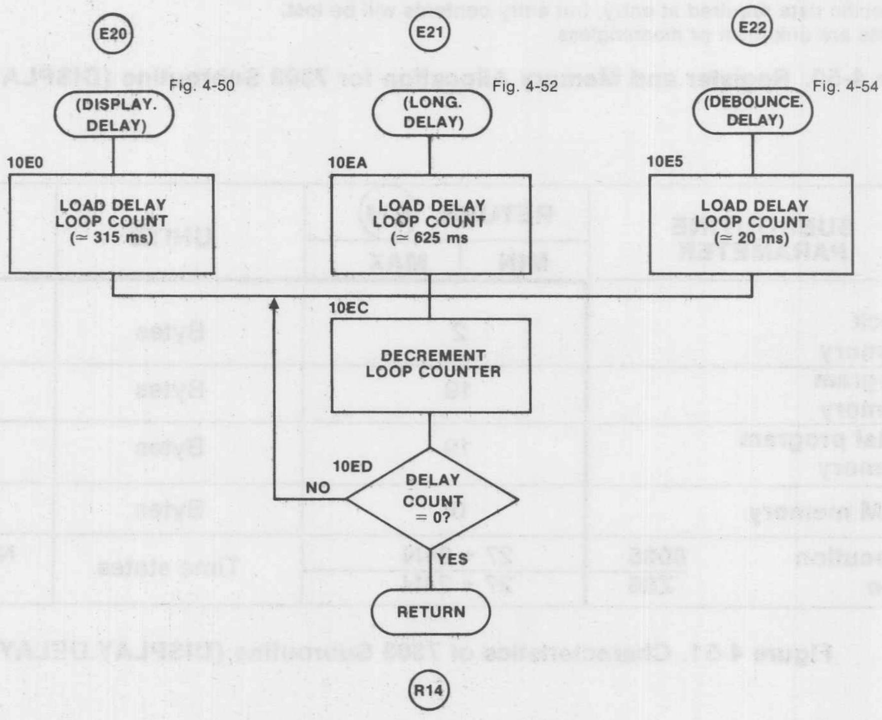


Figure 4-49. Flowchart—Auxiliary Timing Module for the 7303.

( )	Denotes subroutine label
*	Low level active
E/R	Entry/return path identifier encircled

This captive subroutine is used by (BILLBOARD) to pause about 315 ms between display shift operations. (DISPLAY.DELAY) produces an approximate time delay, which depends upon both microprocessor type and clock frequency and is not recommended for other timing applications.

PARAMETER ELEMENT	PARAMETER ADDRESS	ENTRY (E20)	RETURN (R14)	COMMENT
Register	A	XX	??	—
Register	B	XX	00	—
Register	C	XX	00	—
Register	F	XX	??	—

NOTES
1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-50. Register and Memory Allocation for 7303 Subroutine (DISPLAY.DELAY).

SYMBOL	SUBROUTINE PARAMETER		RETURN (R14) MIN	RETURN (R14) MAX	UNITS	COMMENTS
Ns	Stack memory		2		Bytes	—
Np	Program memory		19		Bytes	—
Npt	Total program memory		19		Bytes	—
Nr	RAM memory		0		Bytes	—
Te	Execution time	8085	27 + 24N		Time states	N = Loop count in B,C pair
		Z80	27 + 24N			

Figure 4-51. Characteristics of 7303 Subroutine (DISPLAY.DELAY).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This captive subroutine is used by a demonstration program to pause about 625 ms between display operations.

(LONG.DELAY) produces an approximate time delay, which varies with microprocessor type and clock frequency and is not recommended for other timing applications.

PARAMETER		ENTRY (E21)	RETURN (R14)	COMMENTS
ELEMENT	ADDRESS			
Register	A	XX	??	—
Register	B	XX	00	—
Register	C	XX	00	—
Register	F	XX	??	—

NOTES
1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-52. Register and Memory Allocation for 7303 Subroutine (LONG.DELAY).

SYMBOL	SUBROUTINE PARAMETER		RETURN (R14) MIN	RETURN (R14) MAX	UNITS	COMMENTS
Ns	Stack memory		2		Bytes	—
Np	Program memory		9		Bytes	—
Npt	Total program memory		9		Bytes	—
Nr	RAM memory		0		Bytes	—
Te	Execution time	8085	17 + 24N		Time states	N = Loop count in B,C pair
		Z80	17 + 24N			

Figure 4-53. Characteristics of 7303 Subroutine (LONG.DELAY).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This captive subroutine is used to debounce the key switches, producing a delay in the range of 15-25 ms.

Because the time delay is approximate, varying with microprocessor type and clock frequency, this subroutine should not be used for other timing applications.

PARAMETER ELEMENT	PARAMETER ADDRESS	ENTRY E22	RETURN R14	COMMENT
Register	A	XX	??	—
Register	B	XX	00	—
Register	C	XX	00	—
Register	F	XX	??	—

NOTES
1. For registers not shown, entry contents are not used and remain unaltered at exit.
2. XX means no specific data required at entry, but entry contents will be lost.
3. ?? means contents are unknown or meaningless.

Figure 4-54. Register and Memory Allocation for 7303 Subroutine (DEBOUNCE.DELAY).

SYMBOL	SUBROUTINE PARAMETER		RETURN R14 MIN	RETURN R14 MAX	UNITS	COMMENTS
Ns	Stack memory		2		Bytes	—
Np	Program memory		18		Bytes	—
Npt	Total program memory		18		Bytes	—
Nr	RAM memory		0		Bytes	—
Te	Execution time	8085	27 + 24N		Time states	N = Loop count in B,C pair
		Z80	27 + 24N			

Figure 4-55. Characteristics of 7303 Subroutine (DEBOUNCE.DELAY).

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This program demonstrates a technique for displaying a long message on a display with a limited number of positions, then performs a lamp test. It repeats the message "PRO-LOG 7303" twice, turns on all LED segments, then repeats. See Fig. 4-56. for flowchart.

Requires no initialization except setting the stack pointer.

> **NOTE**
>
> This is an endless loop demonstration program—**not a subroutine.**

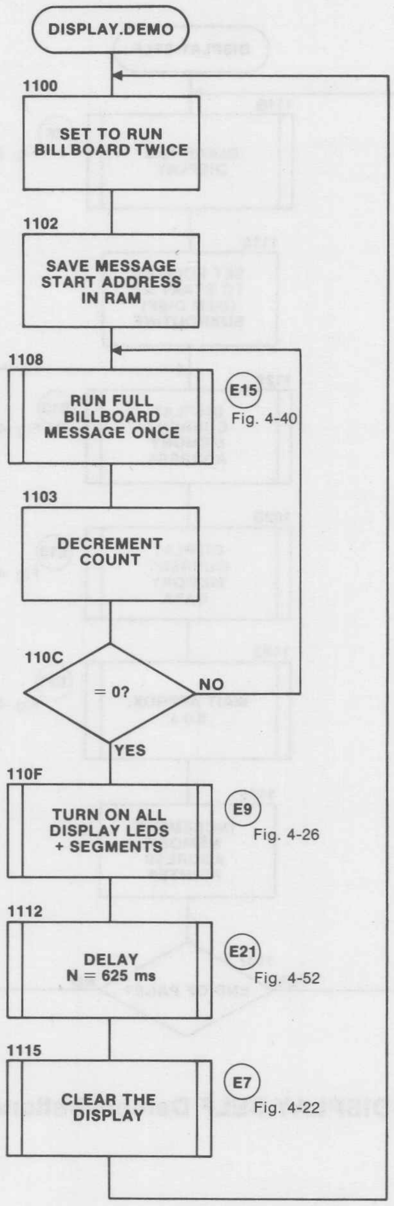


Figure 4-56. Flowchart—DISPLAY.DEMO Demonstration/Test Program for the 7303.

( )	Denotes subroutine label
*	Low level active
E/R	Entry/return path identifier encircled

Displays address/data for every location in memory page 10, which is where the software package's display subroutines are stored. Shows full hexadecimal address (1000-10FF) and hexadecimal data stored at each address, then repeats. See Fig. 4-57 for flowchart.

NOTE

This is an endless loop demonstration program—**not a subroutine.**

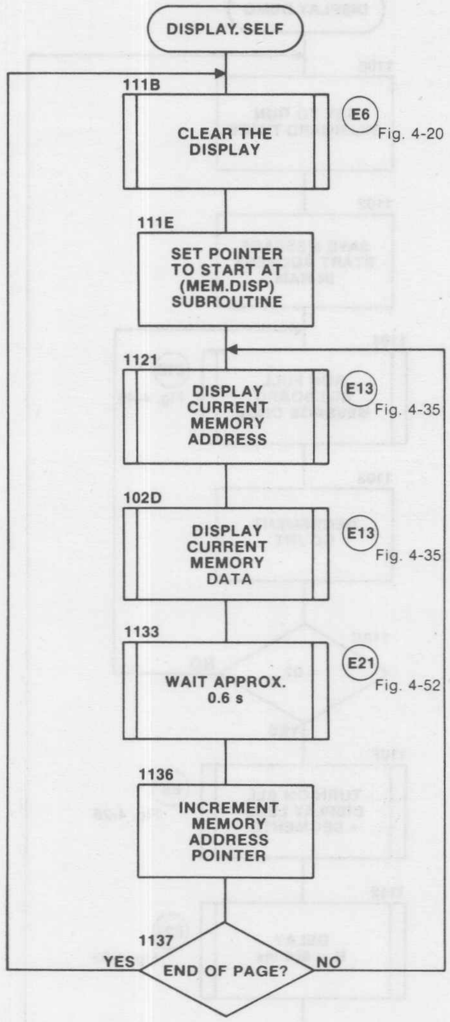


Figure 4-57. Flowchart—DISPLAY.SELF Demonstration/Test Program for the 7303.

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This program reads keystrokes and shifts them across the display, right to left, in the manner of a calculator. The program demonstrates the modular technique of changing display format by manipulating memory rather than rewriting the display routine each time for each new format. The same display subroutine, (MESSAGE), displays the same portion of RAM memory each time, but the memory data is changed each time prior to display. See Fig. 4-58 for flowchart.

NOTE

This is an endless loop demonstration program—**not a subroutine.**

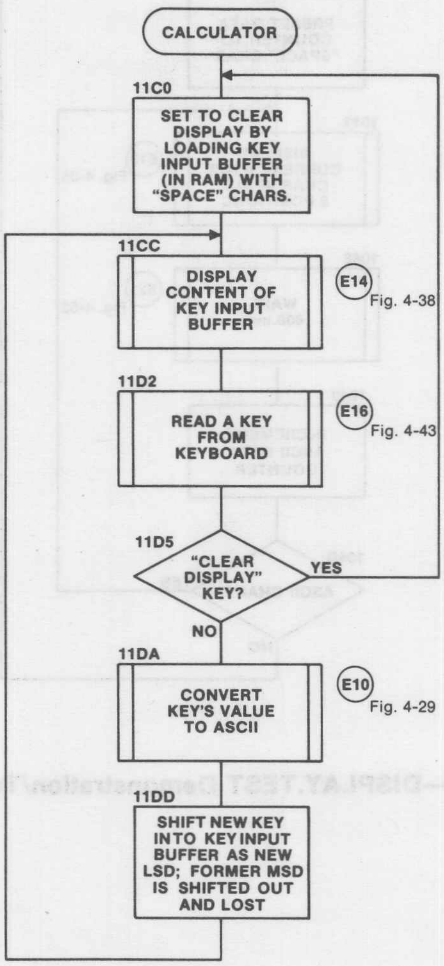


Figure 4-58. Flowchart—CALCULATOR Demonstration/Test Program for the 7303.

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This program allows operator testing of the displays by observing each display position as the program cycles all eight displays through every ASCII character (Fig. 3-3) that can be displayed by the 7303. See Fig. 4-59 for flowchart.

NOTE

This is an endless loop demonstration program—**not a subroutine.**

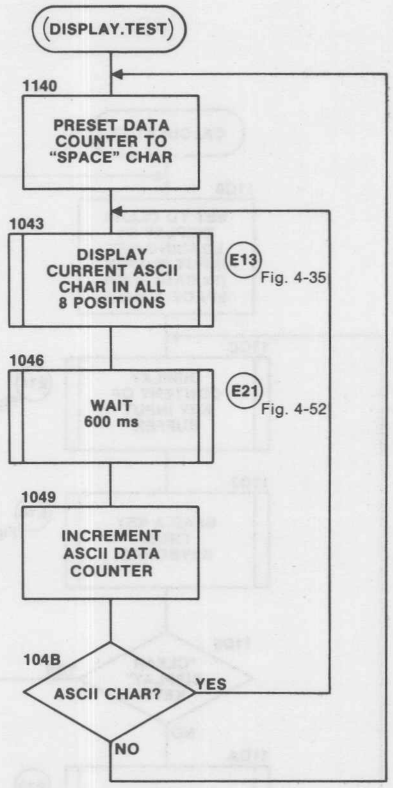


Figure 4-59. Flowchart—DISPLAY.TEST Demonstration/Test Program for the 7303.

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

This program allows an operator to test the 7303's keyboard, by observing that the key value (as labeled on the keys when the card is shipped) appears in the display each time one of the keys is pressed. It does not apply to the RESET key, which resets the system processor (Section 3). See Fig. 4-60 for flowchart.

> NOTE
>
> This is an endless loop demonstration program—**not a subroutine.**

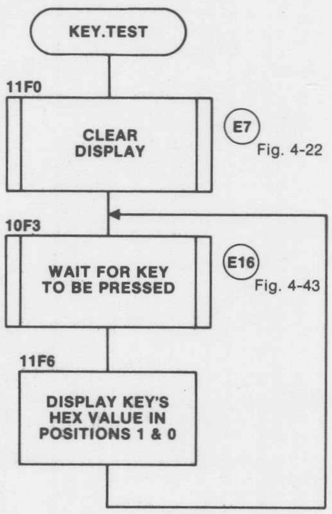


Figure 4-60. Flowchart—KEY.TEST Demonstration/Test Program for the 7303.

( ) Denotes subroutine label
* Low level active
E/R Entry/return path identifier encircled

PRO-LOG CORPORATION — PROGRAM ASSEMBLY FORM

TITLE | DATE

HEXADECIMAL PAGE ADR	HEXADECIMAL LINE ADR	HEXADECIMAL INSTR.	LABEL	MNEMONIC INSTR.	MODIFIER	COMMENTS
10	0 0	7E	(MEM·DISP)	LDAN	H, L	FETCH ASCII CHARACTER FROM MEMORY
	0 1	F6	(DISPLAY)	ORAI		
	2	80			PARITY BIT	OUTPUT ASCII WITH PARITY BIT=1 TO
	3	D3		OPA		DATA PORT
	4	D0			DATA PORT	
	5	00		NOP		
	6	78		LDA	B	
	0 7	E6	(STROBE)	ANAI		OUTPUT DISPLAY POSITION ADDRESS
	8	F7			POSITION·BITS	TO CONTROL PORT WITH WRITE BIT=0
	9	D3		OPA		
	A	D1			CONTROL PORT	
	B	00		NOP		
	C	F6		ORAI		SET WRITE BIT =1
	D	08			WRITE BIT	
	E	D3		OPA		
	F	D1			CONTROL PORT	
10	1 0	00		NOP		SET WRITE BIT=0
	1	EE		XRAI		
	2	08			WRITE BIT	
	3	D3		OPA		
	4	D1			CONTROL PORT	
	5	00		NOP		
	6	C9		RTS		
	7					
	8					
	9					
	1 A	CD	(CLEAR BOTH)	JS		CLEAR CURSORS
	B	80			(CLEAR CURSORS)	
	C	10			—	
	1 D	3E	(CLEAR DISPLAY)	LDAI		WRITE ASCII "SPACE" TO ALL
	E	A0			ASCII "SPACE"	DISPLAYS
	F	C3		JP		
10	2 0	4D			(DISPLAY 8)	
	1	10			—	
	2					
	3					
	4					
	5					
	6					
	7					
	8					
	9					
	A					
	2 B	06	(MESSAGE)	LDBI		START AT DISPLAY POSITION 7
	C	07			POSITION·7	
	2 D	CD	MESSAGE·LOOP	JS		DISPLAY ONE CHARACTER FROM
	E	00			(MEM·DISP)	MEMORY
	F	10			—	
10	3 0	78		LDA	B	
	1	FE		CPAI		CHECK IF POSITION Ø IS
	2	00			POSITION 0	LOADED
	3	C8		RTS	Z1	
	4	23		ICP	H, L	INCREMENT MEMORY POINTER
	5	05		DCB		DECREMENT DISPLAY POSITION ADDRESS
	6	C3		JP		
	7	2D			MESSAGE·LOOP	
	8	10			—	
	3 9	2A	(BILLBOARD)	LDPD	H, L	READ MESSAGE'S START ADDRESS
	A	00			TEXT·START	FROM MEMORY
	B	21			RAM	
	3 C	E5	BILLBOARD LOOP	PSP	H, L	SAVE CURRENT START ADDRESS
	D	CD		JS		
	E	2B			(MESSAGE)	DISPLAY EIGHT CHARACTERS OF MESSAGE
	F	10			—	

100001 2/77

PRO-LOG CORPORATION

PROGRAM ASSEMBLY FORM

HEXADECIMAL PAGE ADR	LINE ADR	INSTR.	LABEL	MNEMONIC INSTR.	MODIFIER	TITLE / DATE COMMENTS
10	4 0	CD		JS		PAUSE APPROXIMATELY 1/2 SECOND
	1	E0			(DISPLAY DELAY)	
	2	10			—	
	3	23		ICP	H, L	SAVE NEXT CHARACTER OF MESSAGE
	4	7E		LDAN	H, L	
	5	E1		PLP	H, L	RESTORE CURRENT START ADDRESS
	6	FE		CPAI		
	7	FF			END·TEXT FLAG	END OF TEXT FLAG NEXT ?
	8	C8		RTS	Z1	
	9	23		ICP	H, L	INCREMENT CURRENT START ADDRESS
	A	C3		JP		
	B	3C			BILLBOARD·LOOP	
	C	10			—	
	4 D	4F	(DISPLAY·8)	LDC	A	START AT DISPLAY POSITION
	E	06		LDBI		
	F	07			POSITION 7	
10	5 0	79	DISP·8·LOOP	LDA	C	DISPLAY ASCII CHARACTER
	1	CD		JS		
	2	01			(DISPLAY)	
	3	10			—	
	4	78		LDA	B	POSITION ZERO LOADED
	5	FE		CPAI		
	6	00			POSITION 0	
	7	C8		RTS	Z1	
	8	05		DCB		
	9	C3		JP		
	A	50			DISP·8·LOOP	
	B	10			—	
	C					
	D					
	E					
	F					
10	6 0	A0	DEMO MESSAGE		SPACE	DEMO MESSAGE
	1	A0			SPACE	LOOK UP TABLE
	2	A0			SPACE	
	3	A0			SPACE	
	4	A0			SPACE	
	5	A0			SPACE	
	6	A0			SPACE	
	7	A0			SPACE	
	8	AA			*	
	9	AA			*	
	A	D0			P	
	B	D2			R	
	C	CF			O	
	D	AD			–	
	E	CC			L	
	F	CF			O	
10	7 0	C7			G	
	1	A7			'	(APOSTROPHE)
	2	D3			S	
	3	A0			SPACE	
	4	B7			7	
	5	B3			3	
	6	B0			0	
	7	B3			3	
	8	A0			SPACE	
	9	A0			SPACE	
	A	A0			SPACE	
	B	A0			SPACE	
	C	A0			SPACE	
	D	A0			SPACE	
	E	A0			SPACE	
10	7 F	FF			END·TEXT·FLAG	END OF TEXT FLAG

100001 2/77

PRO-LOG CORPORATION

PROGRAM ASSEMBLY FORM

TITLE | DATE

HEXADECIMAL PAGE ADR.	LINE ADR.	INSTR.	LABEL	MNEMONIC INSTR.	MODIFIER	COMMENTS
10	8 0	06	(CLR·CURSORS)	LDBI		SET CURSOR BIT PATTERN TO
	1	00			00	ALL ZEROS
10	8 2	3E	(CURSORS)	LDAI		OUTPUT CURSOR BIT PATTERN FOR
	3	0F			POSITIONS 3,2,1,0	POSITIONS 0,1,2,3 TO RIGHT SIDE
	4	A0		ANA	B	OF DISPLAY
	5	D3		OPA		
	6	D0			DATA PORT	
	7	00		NOP		
	8	3E		LDAI		
	9	00			RIGHT SIDE	
	A	CD		JS		STROBE THE DISPLAY
	B	07			(STROBE)	
	C	10				
	D	78		LDA	B	OUT PUT CURSOR BIT PATTERN
	E	0F		RRA		FOR POSITIONS 4,5,6,7 TO LEFT
	F	0F		RRA		SIDE OF DISPLAY
10	9 0	0F		RRA		
	1	0F		RRA		
	2	E6		ANAI		
	3	0F			POSITIONS 3,2,1,0	
	4	D3		OPA		
	5	D0			DATA PORT	
	6	00		NOP		
	7	3E		LDAI		
	8	04			LEFT SIDE	
	9	C3		JP		STROBE THE DISPLAY & EXIT
	A	07			(STROBE)	
	B	10				
	C					
	9 D	E6	(HEX→ASCII)	ANAI		REMOVE BITS 4,5,6,7
	E	0F			0F	
	F	FE		CPAI		IS HEX > 9
10	A 0	0A			0A	
	1	DA		JP	C1	
	2	A7			0-9	
	3	10				
	4	C6		ADAI		NO-HEX CHARACTER A-F, ADD
	5	B7			B7	B7 (YIELDS C1-C6 CODES)
	6	C9		RTS		
	A 7	F6	0-9	ORAI		HEX CHARACTER 0-9 ADD
	8	B0			B0	B0 (YIELDS B0-B9 CODES)
	9	C9		RTS		
	A					
	A B	7E	(MEM/ASCII)	LDAN	H, L	FETCH BINARY USING DATA SOURCE POINTER
	C	E6		ANAI		
	D	F0			F0	RIGHT-ADJUST THE HIGH-ORDER
	E	0F		RRA		HEX DIGIT
	F	0F		RRA		
10	B 0	0F		RRA		
	1	0F		RRA		
	2	CD		JS		CONVERT HEX TO ASCII
	3	9D			(HEX/ASCII)	
	4	10				
	5	12		STAN	D, E	STORE ASCII USING DATA DESTINATION POINTER
	6	13		ICP	D, E	INCREMENT THE DATA DESTINATION POINTER
	7	7E		LDAN	H, L	FETCH BINARY USING DATA SOURCE POINTER
	8	E6		ANAI		ISOLATE THE LOW ORDER HEX DIGIT.
	9	0F			0F	
	A	CD		JS		CONVERT HEX TO ASCII
	B	9D			(HEX/ASCII)	
	C	10				
	D	12		STAN	DE	STORE ASCII USING DATA DESTINATION POINTER
	E	13		ICP	D, E	INCREMENT SOURCE AND DESTINATION POINTER
	F	23		ICP	H, L	

100001 2/77

PRO-LOG CORPORATION — PROGRAM ASSEMBLY FORM

TITLE | DATE

HEXADECIMAL PAGE ADR	HEXADECIMAL LINE ADR	HEXADECIMAL INSTR.	MNEMONIC LABEL	MNEMONIC INSTR.	MODIFIER	COMMENTS
10	C 0	05		DCB		DECREMENT BYTE COUNTER
	1	C2		JP	ZO	
	2	AB			(MEM/ASCII)	
	3	10			—	
	4	C9		RTS		
	5					
	6					
	C 7	CD	(DISP·HEX)	JS		CONVERT HEX TO ASCII
	8	9D			(HEX/ASCII)	
	9	10			—	
	A	C3		JP		DISPLAY ASCII CHARACTER
	B	01			(DISPLAY)	
	C	10			—	
	D					
	C E	79	(DISP·2·N·C)	LDA	C	ISOLATE AND RIGHT-ADJUST
	F	E6		ANAI		BITS 4,5,6,7
10	D 0	F0			F0	
	1	0F		RRA		
	2	0F		RRA		
	3	0F		RRA		
	4	0F		RRA		
	5	CD		JS		DISPLAY HEX
	6	C7			(DISP·HEX)	
	7	10			—	
	8	05		DCB		DECREMENT DISPLAY POSITION ADDRESS
	9	79		LDA	C	
	A	E6		ANAI		ISOLATE BITS 0,1,2,3
	B	0F			0F	
	C	C3		JP		DISPLAY HEX
	D	C7			(DISP. HEX)	
	E	10			—	
	F					
10	E 0	06	(DISPLAY DELAY)	LDBI		LOAD DISPLAY LOOP COUNT
	1	80			~ 31.5 ms	
	2	C3		JP		
	3	EC			PAUSE	
	4	10				
	E 5	06	(DEBOUNCE DELAY)	LDBI		LOAD DELAY LOOP COUNT
	6	0A			~ 20 ms	
	7	C3		JP		
	8	EC			PAUSE	
	9	10				
	E A	06	(LONG DELAY)	LDBI		LOAD LONG DELAY LOOP COUNT
	B	FF			~625 ms	
	E C	0B	PAUSE	DCP	B,C	DECREMENT LOOP COUNTER
	D	78		LDA	B	DELAY COUNT = Ø ?
	E	B1		ORA	C	
	F	C2		JP	ZO	
10	F 0	EC			PAUSE	
	1	10			—	
	2	C9		RTS		
	F 3	06	(LAMP TEST)	LDBI		LOAD CURSOR BIT PATTERN
	4	FF			ALL CURSORS	
	5	CD		JS		DISPLAY ALL CURSORS
	6	82			(CURSORS)	
	7	10			—	
	8	3E		LDAI		LOAD LED BIT PATTERN
	9	FF			ALL LEDS	
	A	D3		OPA		OUTPUT BIT PATTERN TO
	B	D0			DATA PORT	DATA PORT
	C	00		NOP		
	D	C9		RTS		
	E					
	F					

100001 2/77

PRO-LOG CORPORATION

PROGRAM ASSEMBLY FORM

TITLE DATE

HEXADECIMAL PAGE ADR	LINE ADR	INSTR.	LABEL	MNEMONIC INSTR.	MODIFIER	COMMENTS
11	0 0	1E	DISPLAY DEMO	LDEI		SET TO RUN BILLBOARD TWICE
	1	02			COUNT 2	
	2	21		LDPI	H,L	
	3	60			DEMO·MESSAGE	LOAD MESSAGE START ADDRESS IN
	4	10			—	RAM
	5	22		STPD	H,L	
	6	00			TEXT·START	
	7	21			RAM	
	0 8	CD	DEMO LOOP	JS		RUN FULL BILLBOARD MESSAGE ONCE
	9	39			(BILLBOARD)	
	A	10			—	
	B	1D		DCE		DECREMENT LOOP COUNTER
	C	C2		JP	ZO	EQUAL TO ZERO?
	D	08			DEMO LOOP	
	E	11			—	
	F	CD		JS		YES- TURN ON ALL DISPLAY SEGMENTS
11	1 0	F3			(LAMP TEST)	
	1	10			—	
	2	CD		JS		PAUSE 625 MILLISECONDS
	3	EA			(LONG DELAY)	
	4	10			—	
	5	CD		JS		CLEAR THE DISPLAY
	6	1A			(CLEAR BOTH)	
	7	10			—	
	8	C3		JP		REPEAT
	9	00			DISPLAY·DEMO	
	A	11			—	
	1 B	CD	DISPLAY·SELF	JS		CLEAR THE DISPLAY
	C	1A			(CLEAR BOTH)	
	D	10			—	
	E	21		LDPI	H,L	SET POINTER TO START AT (MEM·DISP)
	F	00			(MEM·DISP)	
11	2 0	10			—	
	2 1	4C	SELF·LOOP	LDC	H	DISPLAY THE CURRENT MEMORY ADDRESS
	2	06		LDBI		
	3	07			POSITION 7	
	4	CD		JS		
	5	CE			(DISP·2·IN·C)	(PAGE ADDRESS - A8-A15)
	6	10			—	
	7	4D		LDC	L	
	8	06		LDBI		
	9	05			POSITION 5	
	A	CD		JS		
	B	CE			(DISP·2·IN·C)	(LINE ADDRESS A0-A7)
	C	10			—	
	D	4E		LDCN	H,L	DISPLAY THE CURRENT MEMORY DATA
	E	06		LDBI		
	F	01			POSITION 1	
11	3 0	CD		JS		
	1	CE			(DISP·2·IN·C)	
	2	10				
	3	CD		JS		
	4	EA			(LONG DELAY)	WAIT 625 MILLISECONDS
	5	10			—	
	6	23		ICP	H,L	INCREMENT CURRENT MEMORY ADDRESS
	7	7D		LDA	L	END OF PAGE?
	8	FE		CPAI		
	9	00			00	
	A	C2		JP	ZO	
	B	21			SELF·LOOP	
	C	11			—	
	D	C3		JP		YES — START OVER
	E	1B			DISPLAY·SELF	
	F	11				

100001 2/77

PRO-LOG CORPORATION — PROGRAM ASSEMBLY FORM

PAGE ADR.	LINE ADR.	INSTR.	LABEL	INSTR.	MODIFIER	COMMENTS
11	C 0	21	CALCULATOR	LDPI	H, L	SET TO CLEAR THE DISPLAY BY FILLING
	1	02			KEY BUFFER	THE KEY-INPUT BUFFER (IN RAM)
	2	21			RAM	WITH ASCII "SPACE" CHARACTERS
	3	06		LDBI		
	4	08			COUNT 8	
	C 5	36	ERASE LOOP	LDMI		
	6	A0			ASCII "SPACE"	
	7	23		ICP	H, L	
	8	05		DCB		
	9	C2		JP	ZO	
	A	C5			ERASE LOOP	
	B	11				
	C C	21	DISPLAY BUFFER	LDPI	H, L	DISPLAY THE CONTENTS OF THE
	D	02			KEY BUFFER	KEY-INPUT BUFFER
	E	21			RAM	
	F	CD		JS		
11	D 0	2B			(MESSAGE)	
	1	10				
	D 2	CD	NEW KEY	JS		WAIT FOR A KEY TO BE PRESSED
	3	55			(READ KEY)	
	4	11				
	5	FE		CPAI		CLEAR DISPLAY?
	6	11			"CLEAR" KEY	(KEY #11 IS ARBITRARILY ASSIGNED
	7	CA		JP	ZI	THE "CLEAR DISPLAY" FUNCTION)
	8	C0			CALCULATOR	
	9	11				
	A	CD		JS		NO - CONVERT THE KEY TO ASCII
	B	9D			(HEX/ASCII)	
	C	10				
	D	21		LDPI	H, L	SHIFT INTO THE KEY-INPUT BUFFER
	E	09			KEY BUFFER+7	AS THE NEW LSD (OLD MSD IS LOST)
	F	21			RAM	
11	E 0	06		LDBI		
	1	08			COUNT 8	
	2	4E	SHIFT BUFFER	LDCN	H, L	
	3	77		STAN	H, L	
	4	79		LDA	C	
	5	23		ICP	H, L	
	6	05		DCB		
	7	C2		JP	ZO	
	8	E2			SHIFT·BUFFER	
	9	11				
	A	C3		JP		REPEAT INDEFINITELY
	B	CC			DISPLAY·BUFFER	
	C	11				
	D					
	E					
	F					
11	F 0	CD	KEY TEST	JS		CLEAR THE DISPLAY
	1	1A			(CLEAR BOTH)	
	2	10				
	3	CD	REPEAT	JS		WAIT FOR A KEY TO BE PRESSED
	4	55			(READ KEY)	
	5	11				DISPLAY THE KEY'S HEXADECIMAL VALUE
	6	48		LDC	B	IN POSITION 1 AND 0
	7	06		LDBI		
	8	01			POSITION·1	
	9	CD		JS		
	A	CE			(DISP·2·IN·C)	
	B	10				
	C	C3		JP		REPEAT INDEFINITELY
	D	F3			REPEAT	
	E	11				
	F					

100001 2/77

PRO-LOG CORPORATION — PROGRAM ASSEMBLY FORM

TITLE | DATE

HEXADECIMAL PAGE ADR	LINE ADR	INSTR.	LABEL	MNEMONIC INSTR.	MODIFIER	COMMENTS
11	80	3F			ALL ROWS	
	1	B9		CPA	C	
	2	C2		JP	ZO	ARE THE COORDINATES THE SAME?
	3	64			(DECODE KEY)	
	4	11				
	5	79		LDA	C	YES-CONVERT ROW COORDINATE FROM
	6	0E		LDCI		ONE OF 6 INPUT PORT BITS TO ONE
	7	00			00	OF 6 HEX NUMBERS:
	88	1F	FIND ROW	RRAC		00, 01, 02, 03, 04, OR 05
	9	DA		JP	C1	
	A	90			CONVERT ROW	
	B	11				
	C	0C		ICC		
	D	C3		JP		
	E	88			FIND ROW	
	F	11				
11	90	79	CONVERT ROW	LDA	C	MULTIPLY ROW COORDINATE BY 4 TO
	1	07		RLA		PRODUCE HEX 00, 04, 08, 0C, 10, 14
	2	07		RLA		
	3	4F		LDC	A	
	4	78		LDA	B	CONVERT COLUMN COORDINATE FROM
	5	06		LDBI		ONE OF FOUR OUTPUT PORT BITS TO
	6	00			00	ONE OF FOUR HEX NUMBERS:
	97	1F	FIND COLUMN	RRAC		00, 01, 02, 03
	8	DA		JP	C1	
	9	9F			ADD ROW+COL	
	A	11				
	B	04		ICB		
	C	C3		JP		
	D	97			FIND COLUMN	
	E	11				
	9F	78	ADD ROW+COL.	LDA	B	ADD CONVERTED ROW + COLUMN COORDINATES
11	A0	81		ADA	C	TO PRODUCE KEYS VALUE IN THE
	1	47		LDB	A	RANGE 00-17 HEXADECIMAL
	2	C9		RTS		EXIT
	3					
	4					
	5					
	6					
	A7	3E	(SCAN)	LDAI		STROBE ALL KEY COLUMNS
	8	0F			ALL COLUMNS	SIMULTANEOUSLY
	9	D3		OPA		
	A	D0			DATA PORT	
	B	00		NOP		
	C	DB		IPA		READ ALL KEY ROWS SIMULTANEOUSLY
	D	D0			DATA PORT	
	E	00		NOP		
	F	E6		ANAI		DELETE ROCKER SWITCH BITS AND SET
11	B0	3F			ALL ROWS	Z FLAG: Z=1 IF NO KEY CLOSURES
	1	C9		RTS		EXIT Z=0 IF ANY KEY IS CLOSED
	2					
	3					
	B4	DB	(ROCKER STATUS)	IPA		READ ROCKER SWITCH STATES
	5	D0			DATA PORT	
	6	00		NOP		
	7	17		RLAC		SET CARRY FLAG IF S2 CLOSED
	8	E6		ANAI		SET SIGN FLAG + CLEAR ZERO FLAG
	9	80			SIGN BIT	IF S1 CLOSED
	A	C9		RTS		
	B					
	C					
	D					
	E					
	F					

100001 2/77

PRO-LOG CORPORATION

PROGRAM ASSEMBLY FORM

TITLE

DATE

PAGE ADR	LINE ADR	INSTR.	LABEL	MNEMONIC INSTR.	MODIFIER	COMMENTS
11	40	1E	DISPLAY TEST	LDEI		PRESET ASCII DATA COUNTER TO
	1	A0			ASCII "SPACE"	SPACE CHARACTER
	42	7B	TEST LOOP	LDA	E	DISPLAY CURRENT ASCII CHARACTER
	3	CD		JS		IN ALL EIGHT POSITIONS
	4	4D			(DISPLAY 8)	
	5	10				
	6	CD		JS		WAIT SECONDS
	7	EA			(LONG DELAY)	
	8	10				
	9	7B		LDA	E	INCREMENT ASCII DATA COUNTER
	A	1C		ICE		
	B	FE		CPAI		VALID ASCII CHARACTER?
	C	DF			ASCII "UNDERLINE"	
	D	C2		JP	Z0	(UNDERLINE HAS HIGHEST VALID ASCII
	E	42			TEST LOOP	CODE)
	F	11				
11	50	C3		JP		NO - START OVER
	1	40			DISPLAY TEST	
	2	11				
	3					
	4					
	55	CD	(READ KEY)	JS		ANY KEY CLOSED?
	6	A7			(SCAN)	
	7	11				(NOTE - THIS SEQUENCE ENSURES THAT
	8	C2		JP	Z0	A PREVIOUSLY DECODED KEY HAS BEEN
	9	55			(READ KEY)	RELEASED BEFORE SENSING A NEW KEY)
	A	11				
	B	CD		JS		NO; PAUSE FOR KEY SETTLE TIME
	C	E5			(DEBOUNCE DELAY)	
	D	10				
	E	CD		JS		ANY KEY CLOSED?
	F	A7			(SCAN)	
11	60	11				(NOTE - DUPLICATED SCAN
	1	C2		JP	Z0	SEQUENCE REJECTS NOISE)
	2	55			(READ KEY)	
	3	11				
	64	06	(DECODE KEY)	LDBI		NO: ROTATE A SINGLE BIT ACROSS
	5	80			01	KEY COLUMNS IN KEYBOARD MATRIX
	66	78	DECODE LOOP	LDA	B	UNTIL A KEY CLOSURE IS DETECTED.)
	7	07		RLA		
	8	47		LDB	A	
	9	FE		CPAI		
	A	10			10	
	B	CA		JP	Z1	
	C	64			(DECODE KEY)	
	D	11				
	E	D3		OPA		ANY KEY CLOSED?
	F	D0			DATA PORT	
11	70	DB		IPA		
	1	D0			DATA PORT	
	2	E6		ANAI		
	3	3F			ALL ROWS	
	4	CA		JP	Z1	
	5	66			DECODE LOOP	
	6	11				
	7	4F		LDC	A	YES - SAVE ITS ROW + COLUMN
	8	C5		PSP	B,C	COORDINATES FOR DECODING LATER
	9	CD		JS		PAUSE FOR CONTACT SETTLE TIME
	A	E5			(DEBOUNCE DELAY)	
	B	10				
	C	C1		PLP	BC	COMPARE SAVED + PRESET COORDINATES
	D	DB		IPA		FOR NOISE REJECTION
	E	D0			DATA PORT	
	F	E6		ANAI		

100001 2/77

SECTION 5
Maintenance

Reference Drawings

The schematic (Fig. 5-1) and assembly drawing (Fig. 5-2) in the following pages are included in this manual FOR REFERENCE USE ONLY. They may differ in some respects from the card and documentation that the user receives from Pro-Log.

The schematic and the assembly drawing shipped by Pro-Log with the card are those from which the card was manufactured.

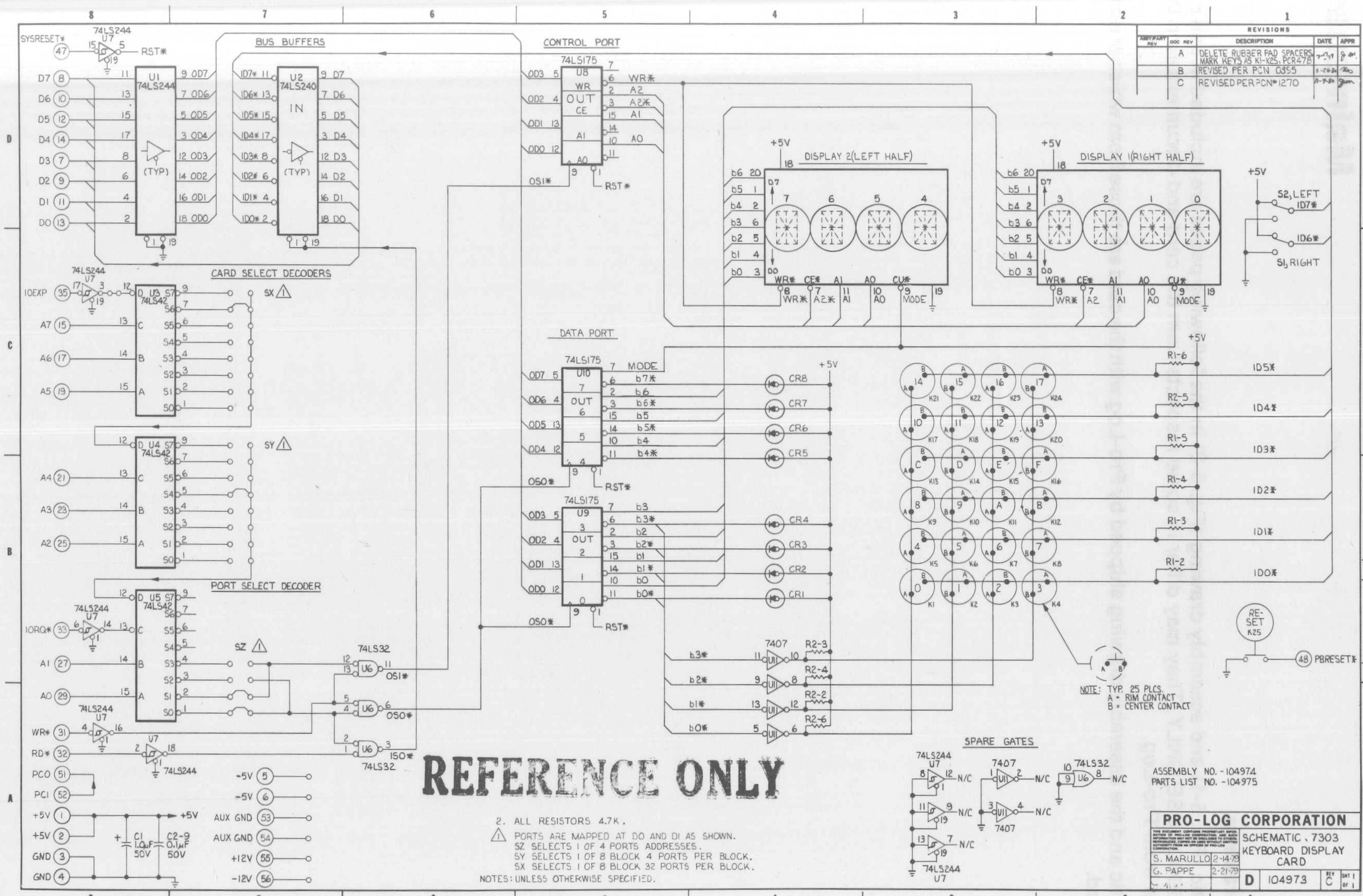


Figure 5-1. Schematic for 7303 (reference only)

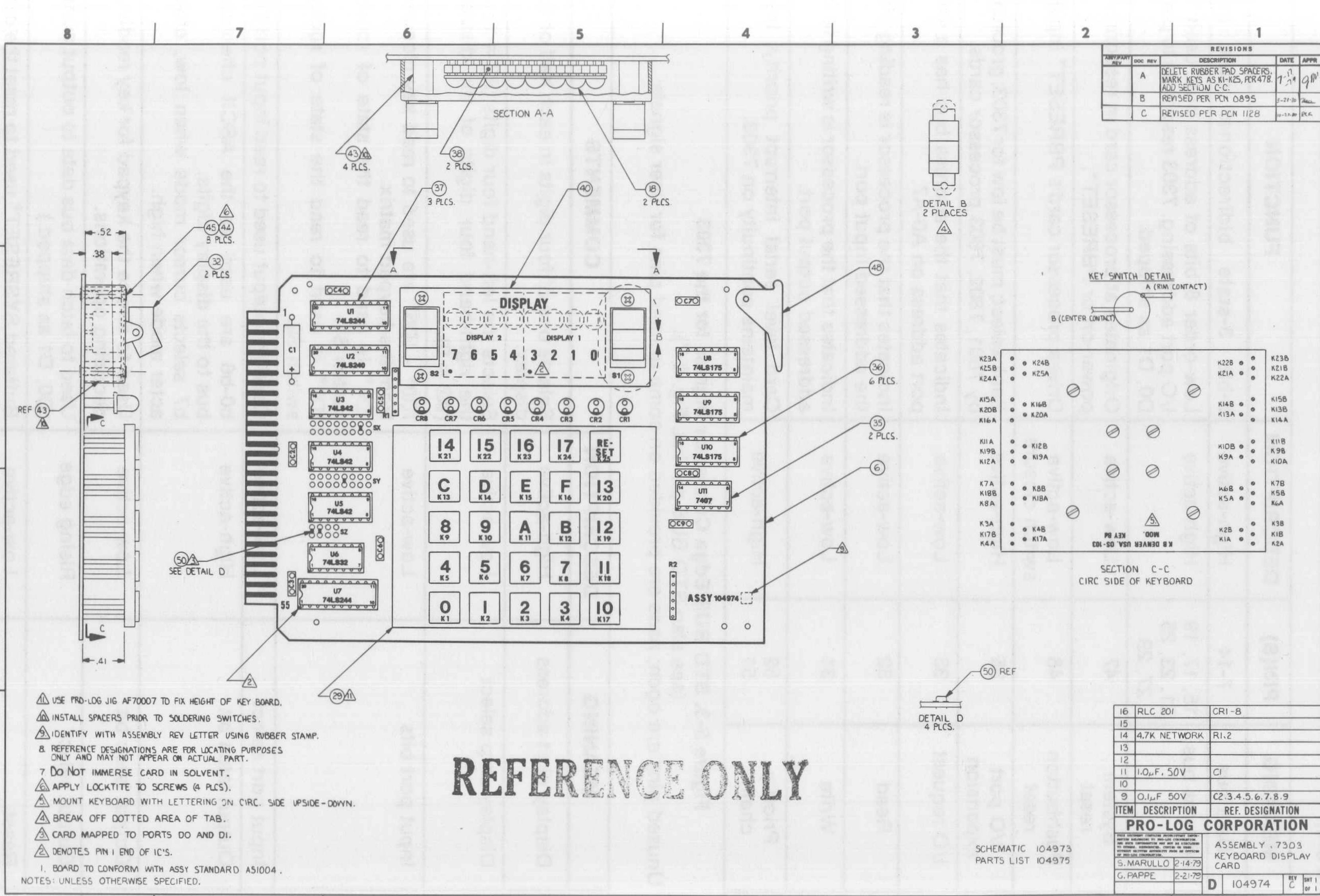


Figure 5-2. Assembly for 7303 (reference only).

Signal Glossary

See Figs. 5-3 and 5-4.

MNEMONIC	MEANING	PIN(S)	DESCRIPTION	FUNCTION
D0-D7	Data bus	7-14	High-active	8-bit, 3-state bidirectional data bus.
A0-A7	Address bus	15, 17, 19 21, 23, 25 27, 29	High-active	Low-order 8 bits of address bus, used for I/O port addressing. 7303 responds to ports D0, D1 as shipped.
SYSRESET*	System reset	47	Low-active	Originates at processor card in response to power-on or PBRESET*.
PBRESET*	Pushbutton reset	48	Low-active switch closure	Drives processor card's PBRESET* input.
IOEXP	I/O port expansion	35	High-active	Bank select; must be low for 7303; grounded by 7801, 7802, 7803 processor cards.
IORQ*	I/O request	33	Low-active	Indicates that the address bus has a valid port address on A0-A7.
RD*	Read	32	Low-active	Indicates that the processor is reading from the addressed input port.
WR*	Write	31	Low-active	Indicates that the processor is writing to the addressed output port.
PCI **PCO**	Priority chain	52 51	High-active	Card level serial interrupt priority; trace maintains continuity on 7303.

Figure 5-3. STD BUS Edge Connector Signals for the 7303
(see also STD BUS pin list, Fig. 2-7).
Note: Unused pins are open; pads are provided on some unused pins for user signals.

MNEMONIC	MEANING	DESCRIPTION	COMMENTS
A0, A1	Display digit address	High-active	Selects one of four digits in each half of the display.
A2, A2*	Display chip select	Low-active	Selects the left-hand four digits (A2 = 1) or the right-hand four digits of the display (A2 = 0).
ID0*-ID7*	Input port bits	Low-active	ID0* - ID5* are used to read key closures from the keypad matrix. ID6* is used to read the state of rocker switch S1. ID7* is used to read the state of rocker switch S2.
IS0*	Input port select	Low-active	Decoder output used to read input port D0.
b0-b7	Output latch bit	High-active	b0-b6 are used as the ASCII character bus to the display digits. b7 selects cursor mode when low, character mode when high.
b0*-b3*	Output latch bit	Low-active	Used to strobe the keypad for key reading/ decoding operations.
OS0*, OS1*	Output select	Rising edge	Used to latch data bus data to output ports. (D0, D1 as shipped.)
RST*	Reset	Low-active	Buffered SYSRESET* used to reset the output ports and the binary LED display.

Figure 5-4. Internal 7303 Signals.

Keyboard Label Replacement

To change a keyboard label, grasp the clear keyswitch cover at the top and bottom edges with the fingernails of your thumb and index or middle finger, then pull directly out and away from the keyboard. The clear plastic cover will snap free from the keyswitch, exposing the legend area below. The legend may then be replaced or covered over by a new legend such as those legends provided by Pro-Log or appropriate sized legends provided by the customer. Replace the clear plastic cover on the keyswitch.

> **WARNING**
>
> Do not expose the 7303 keyboard to fluxes, solvents, cleaning solutions, or their fumes.

Keyboard Disassembly

To replace an individual key, take out the eight slotted screws located underneath the keyboard. Holes in the circuit card provide access to these screws from the card's rear. When the screws are removed, the keycaps fall free with the cover for easy removal. When re-assembling the keyboard, use a mechanical screw starter.

Special Parts

The following parts (Fig. 5-5) may not be readily identifiable by markings on the parts themselves. Should the user desire to obtain these parts from local sources other than Pro-Log, the following information is given concerning their manufacture:

PART	PRO-LOG PART NUMBER	MANUFACTURER	MANUFACTURER'S PART NUMBER
Alphanumeric Display	902085	LITRONIX	DL-1416
Keyboard	902084	K. B. DENVER	MOD 25-01-02-00
Rocker Switch	901359	C&K COMPONENTS or JBT SUBMINIATURE SWITCH	7810 MT77

Please note that replacement of parts by the customer may VOID THE PRO-LOG WARRANTY. Pro-Log assumes no responsibility for the continued availability of these parts.

Figure 5-5. Special Parts for 7303.

Return for Repair Procedures

Domestic Customers:

1. Call our factory direct at (408) 372-4593, and ask for CUSTOMER SERVICE.
2. Explain the problem and we may be able to solve it on the phone. If not, we will give you a Customer Return Order (CRO) number.

   Mark the CRO number on the shipping label, packing slip, and other paperwork accompanying the return. We cannot accept returns without a CRO.
3. Please be sure to enclose a packing slip with CRO number, serial number of the equipment, if applicable, reason for return, and the name and telephone number of the person we should contact (preferably the user), if we have any further questions.
4. Package the equipment in a solid cardboard box secured with packing material.

   CAUTION: Loose MOS integrated circuits, or any product containing CMOS integrated circuits, must be protected from electrostatic discharge during shipment. Use conductive foam pads or conductive plastic bags, and never place MOS or CMOS circuitry in contact with Styrofoam materials.
5. Ship prepaid and insured to:

   Pro-Log Corporation
   2411 Garden Road
   Monterey, California 93940

   Reference CRO #______________.

International Customers:

Equipment repair is handled by your local Pro-Log Distributor. If you need to contact Pro-Log, the factory can be reached at any time by TWX at 910-360-7082.

Limited Warranty: Seller warrants that the articles furnished hereunder are free from defects in material and workmanship and perform to applicable, published Pro-Log specifications for one year from date of shipment. This warranty is in lieu of any other warranty expressed or implied. In no event will Seller be liable for special or consequential damages as a result of any alleged breach of this warranty provision. The liability of Seller hereunder shall be limited to replacing or repairing, at its option, any defective units which are returned F.O.B. Seller's plant. Equipment or parts which have been subject to abuse, misuse, accident, alteration, neglect, unauthorized repair or installation are not covered by warranty. Seller shall have the right of final determination as to the existence and cause of defect. As to items repaired or replaced, the warranty shall continue in effect for the remainder of the warranty period, or for ninety (90) days following date of shipment by Seller or the repaired or replaced part whichever period is longer. No liability is assumed for expendable items such as lamps and fuses. No warranty is made with respect to custom equipment or products produced to Buyer's specifications except as specifically stated in writing by Seller and contained in the contract.

APPENDIX A
Front Panel Mounting of 7303 Card
PLAN 131

Introduction

The 7303 is designed as a direct interface to the STD BUS. If you mount the 7303 outside the STD BUS card rack, do not connect it directly to the STD BUS through a long cable. Such a connection increases backplane capacitance and cross coupling, and results in excessive crosstalk, noise, and generally degraded performance.

Instead, connect the 7303 to the end of I/O port lines. This type of connection (Fig. A-1) requires more program involvement, but it avoids the problems associated with transmitting fast processor signals over a long cable. In this mode, a TTL I/O card with 3-state I/O lines provides the signals needed to control the 7303 in place of the direct STD BUS drive. The program generates these signals by executing short instruction sequences instead of the single read and write instructions used in I/O mapped operation.

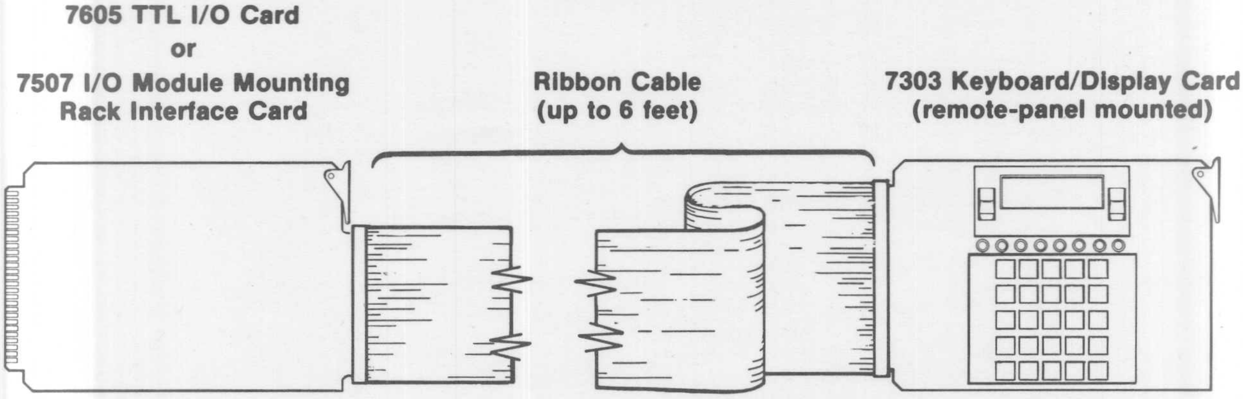


Figure A-1. Cable Connection when Operating the 7303 as an I/O Load.

Remote 7303 Drive Via I/O Lines

When driving the 7303 via I/O lines, the 7303's address decoder circuitry is not used, since the program, instead of the usual hardware, controls card selection. Only address line A0 is retained to select between the two sequential port addresses on the 7303. In addition, the IORQ* and IOEXP lines are not used, since the RD* and WR* signals alone can maintain full card control.

Using either Pro-Log's 7507 module mounting rack interface, 7605 general purpose TTL I/O card, or equivalent I/O card with bidirectional I/O capability, connect the 7303's edge connector as follows:

1. Ground address lines A1 throu A7 (edge connector pins 15, 17, 19, 21, 23, 25, 27, to pins 3, 4); move address jumpers to X0, Y0, Z0, and Z1.
2. Ground IORQ* and IOEXP (edge connector pins 33 and 35 to pins 3, 4).
3. Connect a 3-state I/O port (8 lines) to the 7303's data bus d0-d7 (edge connector pins 13, 11, 9, 7, 14, 12, 10, 8), maintaining one-to-one bit significance for programming convenience.
4. Connect four output-only lines to the 7303's A0, RD*, WR*, and SYSRESET* lines (edge connector pins 29, 32, 31, and 47, respectively). Note that these lines are always outputs from the 3-state I/O card and must remain driven at all times for correct 7303 operation. Do not allow these lines to float unless pull-up resistors are connected.

> NOTE
>
> In steps 3 and 4 above, the interface cable to the 7303 should consist of ribbon cable with alternating ground-signal-ground, or multiple twisted pairs consisting of signal/ground in each pair. Limit cable length to 6 feet (2 meters).

5. Connect +5V ±5% and logic ground to edge connector pins 1, 2, and 3, 4, respectively, via a twisted pair of 18-gauge wires or larger.
6. (Optional). If the 7303's reset pushbutton is to be functional, connect the card's pin 48 to STD BUS trace pin 48.

To program the 7303 as an I/O load, substitute instruction **sequences** for the single read/write instructions normally used. These sequences are as follows:

WRITE Sequence

Select A0 = 0 or 1 (select the 7303's data or control output ports).
Write output data to d0 through d7 at the 7303.
Set WR* = 0.
Set WR* = 1.
Float the data bus drivers.

READ Sequence

Set A0 = 0 (select the 7303's input port).
Set RD* = 0.
Read the input data from 7303's d0 through d7.
Set RD* = 1.

Panel Mounting

The recommended cutouts for mounting the 7303 in a panel are detailed in Fig. A-2. Mount the 7303 in panel stock of up to 0.125-in. thickness, using the four mounting holes provided on the card. The display bezel is recessed approximately 0.375 in. below the keycaps and binary LEDs. This recessing allows for beveling around the display cutout, while the keys and LEDs protrude from the panel front (Fig. A-3).

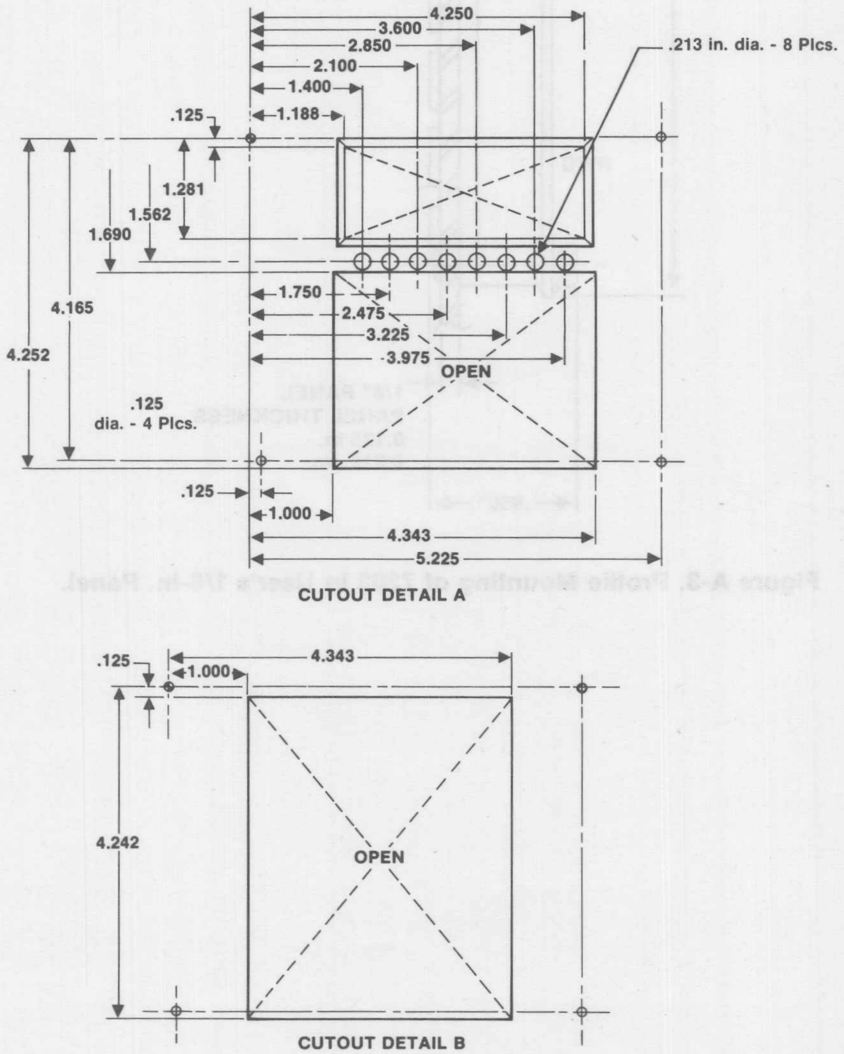


Figure A-2. Cutout Details of 7303 Panel-Mounting (dimensions in inches).

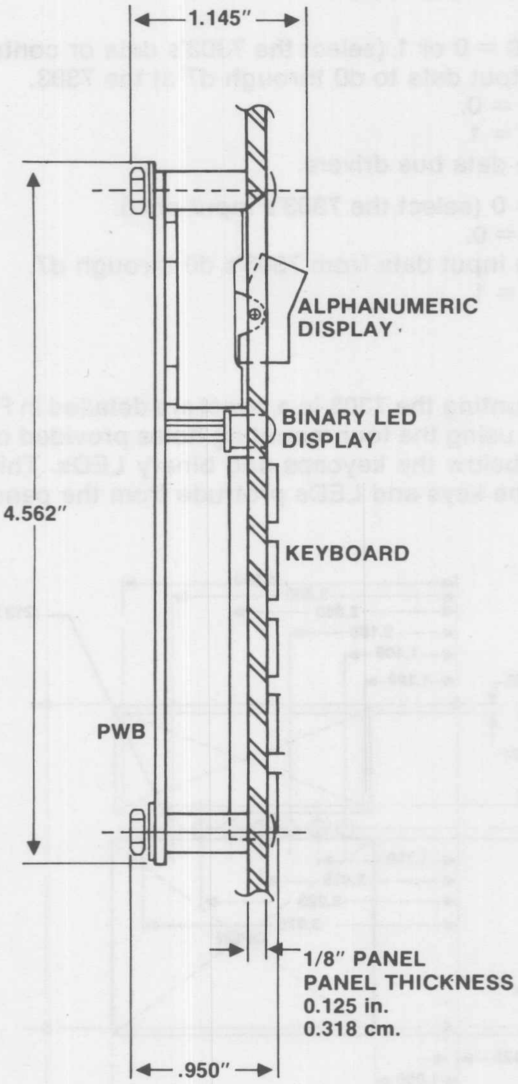


Figure A-3. Profile Mounting of 7303 in User's 1/8-in. Panel.